A MUSICOLOGICAL RESEARCH TOOL: AN EXPERT SYSTEM SOLUTION FOR SMALL PROJECTS

by

JEANNINE STAFFORD INGRAM

B.S., University of North Carolina-Greensboro, 1982

A REPORT

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Computing and Information Sciences

KANSAS STATE UNIVERSITY
Manhattan, Kansas

1989

Approved by:

Elizabeth A. Unger

LD
2662
.R4
CMSC
1989
I54
C.2

A11208 317525

1. OVERVIEW

1.1 Introduction

The purpose of this project is to provide a research tool for musicologists and small music archives in need of cataloging assistance. Development of a tool to assist in this specialized task requires that two areas of musicological research be addressed: incipit matching and document handling.

Typically, musicologists must search through vast quantities of data attempting to gather sufficient information to reconstruct the scope of a collection of music, to ascertain the creative output of a particular composer or group of composers, or to seek all of a single compositional type dating from a specific period. For example, dissertations have been written which survey vocal compositions based exclusively on the "Salve regina" text dating between 1410 and 1550. Similarly, music archives routinely allocate years of staff time in an effort to determine the interconnections between manuscripts belonging to a collection of music developed during a fifty-year span, attempting to link written

documents with musical manuscripts and correctly identify the actual composers of the compositions within the collection. Much of the success of a musicological project is dependent upon the talent of the researchers who sort through this frequently incomplete and often misleading collection of data and, by virtue of their expertise in the area and their knowledge of peripheral data, ultimately reconstruct the most likely solution to the puzzle.

Music historians must deal with a variety of documents in order to complete a given project. Generally, these include letters, music manuscripts, published music, unpublished textual manuscripts, and books. While many documents can be referenced by conventional methods used by researchers in other disciplines, music manuscripts and pre-1900 printed music create problems in the area of erroneous ascriptions and rhythmic, melodic, or key variances introduced by different copyists or publishers. These discrepancies, along with the simple need to derive all known sources of a composition, provide the impetus for the primary thrust of this project -- incipit matching.

In addition to the benefits to be gained from an expert system capable of incipit matching, the process of musicological research can also derive obvious benefits from a database system. The database structure would provide storage and manipulation of data relating to a single project while also assisting in the analysis of this data via the incorporation of rules and inference provided by historical musicologists.

This paper seeks to document a project whose objective is to provide a research tool which will suffice for a variety of musicological projects. The primary emphasis of the system is to store the knowledge of a professional musicological expert in a high-level representation and then to use this information to assist, via incipit matching, in the attribution of compositions to probable composers.

The literature review which follows this section addresses expert systems, the relation of expert systems to musicological studies, and the role of database systems in this arena. The report then continues by describing the development of the musicological tool. Chapter 2 discusses the process of musicological research, describing a typical research project. Chapter

3 presents the requirements of the system being offered as a research tool. This is followed by a description of the design of the system in Chapter 4. Chapter 5 documents implementation and testing of the software. The report concludes with a presentation of conclusions and suggested future enhancements in Chapter 6.

1.2 Literary Review

Resources germane to this project fall into three general categories: (1) database design; (2) artificial intelligence and its application in expert systems; and (3) the integration of expert systems into the field of musicological research. Of these, both expert systems and the use of expert systems as a musicological research tool are relatively new areas of study: indeed, little or no work exists which harnesses the power of an expert system for the benefit of musicology. Database design, on the other hand, is a topic amply covered in computer science circles.

1.2.1 Database Design

A database system is described as a computerized record-keeping system -- that is, a system whose overall purpose is to maintain information and to make that information

available on demand. The sheer volume of data which must be stored, retrieved, manipulated, and referenced as part of any musicological research project makes such a structure essential to the success of the musicological research tool. C. J. Date, in his discussion of why one would choose to employ a database system, summarizes the usefulness of a database system over paper-based methods of record-keeping. He lists (1) compactness, (2) speed, (3) less drudgery, and (4) currency as the most obvious advantages [Da87]. Clearly, the attribute of compactness is welcome relief to the musicological researcher who not only must deal with vast quantities of data, but also must generate equally large quantities of intermediate-level data. The ability to store findings in an easily-retrievable, non-paper form would greatly reduce the workload of musicologists. The attributes of speed, less drudgery, and currency are equally desirable to the music historian whose task neccessitates sifting through an untold numbers of documents in the course of even the smallest research project.

Organization of data within most database systems developed within recent years has been based on relational models. The distinguishing features of relational databases are: (1) the data is perceived by

the user as tables and (2) the operators at the user's disposal for data retrieval or manipulation are operators which generate new tables from old [Da87]. Such a relational model is an ideal storage method for the project currently being described: the whole intent of the system is to create new tables of probable matches or close approximations of incipits based on the use of AI technology.

1.2.2 Expert Systems

In some ways, expert systems seem to be especially suited to musicological research. Certainly, if one accepts Christopher F. Chabris's description of the essential characteristic of an expert system, this is true. Chabris states that expert systems are intended to solve problems in difficult, unstructured domains where knowledge and judgement rather than procedure guides the expert's reasoning [Ch87]. As noted earlier, historical musicology may be among the most unstructured domains since it is, in fact, an interdisciplinary field which requires in-depth knowledge in several areas of study and where many of a scholar's decisions are based on educated intuition rather than simple procedure.

Even clearer justification of the application of expert systems in the field of musicology comes with an examination of Richard Forsyth's description of what makes a suitable candidate for an expert system. [Fo86] states:

> To decide whether a proposed application is likely to benefit from the knowledge-based approach to systems design, you should consider the following questions: (1) Is the problem diagnostic? (2) Is there no established theory? (3) Are the human experts rare? (4) Is the data "noisy" or uncertain.

Again, the field of musicological research can provide resoundingly positive responses to each of these questions.

The design intent of expert systems is to assist users with domain-specific problem-solving expertise by encoding the same problem-solving heuristics that are used by the human experts. Previously-developed successful systems have been employed in a wide spectrum of problem areas including chemical spectrogram analysis (Buchanan & Fergenenbaum, 1978), medical diagnosis (Shortliffe, et al., 1979), mineral exploration (Duda, et al., 1979), genetic engineering (Steefile, 1980), and computer systems configuration (McDermott, 1980). Lehner and Barth [Le85] emphasize that it is clear from the

variety and success of these systems that implementing expert systems on microcomputers is a serious option for potential users of AI technology. Additionally, the use of microcomputers guarantees other factors such as low cost, availablility, and transportability: all these features are essential in building a viable musicological research tool as well.

If one accepts the premise that an expert system can serve as a valid musicological research tool, one must address the concept of expert systems in general. Numerous overviews of the topic are available for study. Parker [Pa88] presents a concise overview of expert systems, citing five elements of an expert system:

(1) facts in the form of rules

(2) an inference engine (procedural knowledge)

(3) an explanation generator

(4) a knowledge acquisition engine

(5) a natural language processor.

Other scholars vary slightly from this format. Forsyth [Fo86] cites four essential components of a fully-fledged expert system:

(1) the knowledge base

(2) the inference engine

(3) the knowledge acquisition module

(4) the explanatory interface.

According to Forsyth, all four modules are critical. A knowledge-based system may lack one of them, but a truly expert system should not. More recently, Vesonder [Ve88] states that rule-based expert systems consist of three components:

(1) a database, more commonly referred to as the working memory

(2) a set of rules, the rule memory

(3) the interpreter, referred to as the inference mechanism.

1.2.3 Musicological Resources

Perhaps due to the "art" involved in musicological research, i.e., the drawing together of knowledge from so many disciplines which must occur in the research process, musicologists have not turned to computers as a true research tool. Robert Skinner, in a review of micro-computer use in music libraries, ascertains that the largest category of music software consists of programs providing computer-assisted instruction. The second largest category of software is that devoted to the

production of actual music: composition, music printing, and musical performance [Sk88].

Clearly, musicologists and music librarians recognize the potential of the computer as a storage medium. In fact, Skinner [Sk88] notes that "CD-Rom and its successors seem natural mediums for encoding composers' complete works so that they can be subjected to various sorts of software-implemented analysis, such as searching for all occurrences of a theme.... Although no music reference works have yet to be published in this format, these may become available in the not-too-distant future, enabling us to search, say, a digital Bach thematic catalog by melodic incipit."
Thus the thought is there, but is yet to be implemented. Instead, the world of music tends to use the computer to assist in teaching music theory or in generating music itself.

While an expert system capable of executing a search and compare mission among musical incipits is yet to be developed, the power of expert systems has been harnessed to invent musical incipits in the style of a particular composer. Ranada [Ra89] describes a newly-completed SUNY/Buffalo PH.D. dissertation by IBM researcher Kemal

Ebciogh, "Report on the CHORAL Project: An Expert System for Harmonizing Four-Part Chorales" which automates the harmonization of chorale melodies in the style of Johann Sebastian Bach. However, even this project deals more with the composition of music than with the historical interpretation of music.

The idea of using computers as a musicological research tool came about in the 1960's and 1970's. Recently, Walter Hewlett and Eleanor Selfridge-Field [He86] initiated a directory listing all known computer-assisted research in musicology in 1986 expressly "...for the purpose of determining what had become of efforts initiated in the 1960's and 1970's to use computers in the field of musicology." They, too, discovered that the use of computers was confined to:

(1) printing music

(2) computer-assisted instruction

(3) cataloging efforts.

Thus the integration of expert systems into the field of musicological research is, for the most part, an unexplored area.

The value of databases as a musicological research tool has been recognized for some time. Yet this, too, is an area besieged by problems. Many scholars point to the lack of standardization and consolidation which limits the usefulness of the research of their colleagues. Both Charnasse [Ch84] and Drummond [Dr84] address the problems encountered by researchers which relate to lack of formalization of database design, structure, and implementation. However, none who attack this problem can offer a clear solution since the use of databases in musicological research may involve diverse music notations, as well as diverse literary and historical references for which no clear-cut standard exists. Warren Hultberg [Hu84], documenting his study of relationships among Spanish music scores of the 16th and 17th centuries, most succinctly summarized both the need and dilemma when he wrote:

> As the study progresses, the pressing need for the development of appropriate data bases becomes increasingly apparent. Generally accepted designations of stylistic features, theory as related to practice, and the basic conventions of the period as seen in the sources, tend to be overly simplified and based on limited material. The development and utilization of data bases designed for the study of the sources considered here, offer possibilities for a greatly enhanced understanding of not only relationships among the Spanish sources, but also their relationships to other literature and repertoires. Quantification of certain aspects of study contributes

to qualitative understanding of the repertoire and its underlying theories.... Appropriate software for comparative studies dealing with analytical problems, practical and theoretical positions, thematic derivation and indexing, is not readily available... for less expensive, micro systems.... Availability of moderately-priced micro systems enhances opportunities for information sharing among scholars in a manner heretofore hardly imaginable, but considerable software development and modification are required if meaningful project advancement is to be achieved.

1.3 Summary

The objective of this project is to produce an online, menu-driven system which will allow research musicologists to build and query a database of information pertinent to their current project. A research tool of this nature has the potential to significantly reduce research time for scholars and students working in musicological research. In an effort to build a more marketable product, implementation of the project is in a format readily available to the personal computer user-community. Ease of use is also a key factor in the development of the product, since the ultimate goal is to decrease the work load of the researcher via improved access to material and computerized decision-making.

2. THE PROCESS OF MUSICOLOGICAL RESEARCH

2.1 Musicology as a Discipline

Musicology, the scholarly study of music, was first introduced as a discipline by Friedrich Chrysander in 1863. Chrysander's intent was to encourage the same high standards of accuracy in the historical study of music as in other areas in the natural sciences and humanities [Ap72]. In the past 125 years, this discipline has become one of sound historical research techniques. Like all historical research, it requires that the professional have a broad knowledge base. A second requirement is that music historians be able to organize and assimilate vast quantities of data, all of which may play some part in the final research product.

Relatively speaking, musicology is a young discipline within the humanities: as such, it is frequently unknown or misunderstood by the masses. For this reason, the following description of typical musicological research projects is provided as a means of establishing the environment of the proposed musicological research tool.

2.2 The Typical Musicologist

Scholars in historical musicology are generally trained in languages, music theory and/or physics, history, and specialized topics in music history. Another area of musicology, that of ethnomusicology, requires additional training in non-Western music theory and culture. Within the arena of historical musicology, researchers frequently specialize in the music of a particular era, i.e., Medieval, Renaissance, Baroque, Classical, Romantic, and Modern, in much the same manner as the art historian. Further specialization will limit the musicologist's area of study to the works of a single composer or group of composers, musical compositions of one genre or style, or compositions from a specific geographic area. Thus, a research project can be devoted to vocal settings of the "Salve regina" text attributed to Franco-Flemish composers between 1425 and 1550. Or a multi-membered research project may deal with the reconstruction of the instrumental repertory of amateur collegii musica functioning in American communities between 1770 and 1830.

2.3 The Research Project

A review of the repertory reconstruction project cited in the previous paragraph provides considerable insight into the musicological research process. It also emphasizes the potential value of a mechanized research tool such as that proposed by this project. Between 1973 and 1982, the National Endowment for the Humanities funded a project which employed four full-time research musicologists, two typists, and supporting staff. All worked toward the reconstruction and documentation of the repertory of the Salem [NC] **Collegium Musicum** and the Bethlehem [PA] Philharmonic Society. Both communities were insular Moravian communities whose strong church ties guaranteed the preservation of a vast archive of letters, diaries, musical compositions, manuscripts, and printed documents which would allow an accurate reconstruction of the musical repertory of these performing groups during America's formative years.

The project was undertaken without computerized assistance, instead using card files and the skill of the researchers to recognize and relocate within those files

any matching incipits, paper watermarks, or references to performances. Since there was no centralized storage medium, problems of conflicting attributions and lack of proper identifications were actually compounded by multiple researchers dealing with a single topic. In short, the project could have been completed much sooner and in a far more thorough manner with the assistance of a mechanized research tool.

2.4 Research Procedures

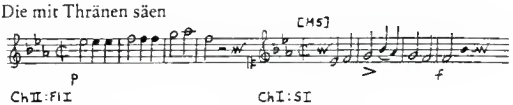
A typical work day for a researcher assigned to this project included hours spent examining music manuscripts and copying pertinent information on index cards. These abbreviated references included:

1. Call number: a sequential number for subsequent referencing
2. Identification: the composer attribution, composer's dates, title(s)
3. Musical incipit
4. Material quoted from title pages, captions
5. Information about the composition: tempo, key, length
6. Inventory of parts or scoring

7. Miscellaneous comments

8. Cross-references.

Illustration 2.1 shows a page of the final index produced for the catalog of compositions belonging to the relatively small Lititz Congregation Collection [St81].

194 BLÜHER, [AUGUST] (–1839)
Die mit Thränen säen

N° 194. . . . 2 *Chörig*. Cat, pencil: *Begräbn*.
Larghetto. E-flat maj. 96m.
Pts: Ch I: S I,II,A;Vln I,II;Vla;Vcl. Ch II: T I,II,B;Fl I,II;Vln I,II;Vla;Vcl. Org.

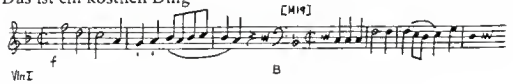
195 REISSIGER, [KARL GOTTLIEB] (1798–1859)
Das ist ein köstlich Ding

T p: N° 195. . . . *Kapellmeister Reissiger*./[pencil] (*in Dresden*)/*für die Brüder-Gemeine*/*componirt*.
Moderato ma poco. F maj. 156m.
Pts: S(10),A(3),T(2),B(3);Vln I,II,III;Vla;Vcl;Org.
Vln III and vla pts identical.

Illustration 2.1

In addition to cataloging, researchers spent hours conversing with colleagues who might have data in the form of other attributions or references which pertained to a special area of interest. Still more time was spent scouring written documents from the era which might contain references to composers, performances, or compositions: such references were, again, meticulously copied on index cards, indexed by date, and cross-referenced by topic.

2.5 Research Product and the Proposed Research Tool

Periodically, an individual researcher would feel confident enough about the data to produce a research paper or article. At that time, the musicologist would search the index file for information which might support a hypothesis evolved during the research process. At this point, as well as the initial information-gathering stage, a musicological research tool could have been of untold assistance. If available, the research tool could identify and locate any duplicate compositions by providing a mechanized search of incipits which would reveal conflicting attributions and multiple copies. A

topical search of written documents could organize literary references and assist in the actual structuring of the researcher's final document. All these are necessary steps in the preparation of an article and all were done without mechanized assistance for the project previously described.

3. REQUIREMENTS

3.1 Introduction

Research musicologists have recognized for quite some time that the computer should be a valuable tool; however, they somehow have failed to develop an integrated approach to the use of this powerful research assistant. Clearly, all who have worked for months amassing data for a research paper, attempting to document and match partially-attributed or incorrectly-attributed compositions or gathering all the necessary literary references for a project, would welcome mechanized relief. Among the many areas within the humanities, musicology perhaps would most benefit from a single, integrated system capable of assisting in the completion of a multi-faceted project.

This system should consist of a user interface capable of providing a straightforward method of accessing the system. As a rule, musicologists are not particularly familiar with automated systems, thus requiring that the proposed system provide a clear set of instructions. In addition to the user interface, the system should include a minimum of four additional features:

(1) an expert system for incipit matching

(2) a graphics capability for entry of these incipits into the system as well as for subsequent displays to the user

(3) a file-handling interface to link the graphics frontend to the database management system

(4) a database management system.

The objective of this project does not include the graphics interface. This will be the design effort of another Kansas State University master's project [Ha89]. Nevertheless, the two designs are closely integrated into a complete system as shown by Illustration 3.1.

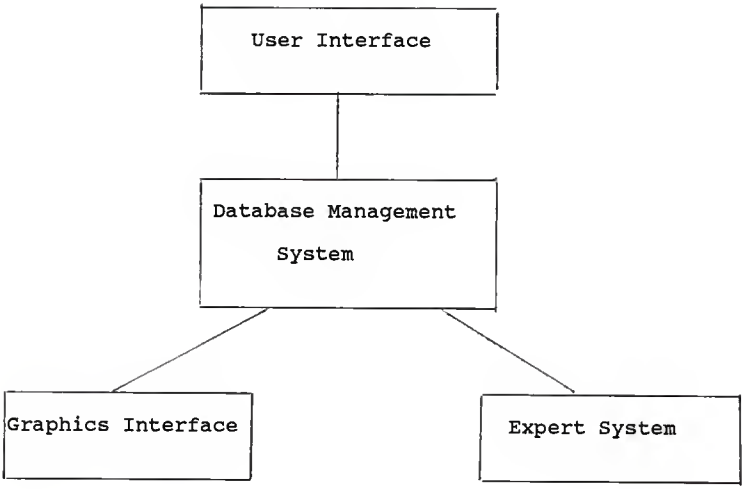


Illustration 3.1

A description of each of these elements, as well as general requirements of the research tool follows.

3.2 General Requirements

The general research requirements of the musicological research tool are three-fold. It should be economical.

Musicologists, like other scholars working in the humanities, are not highly-paid despite the fact that their training must be extensive and broad in scope. For reasons of economy and convenience, the system must run on a personal computer. In addition, personal computers are more familiar to the proposed user since they are frequently employed as word processors by this group of scholars.

A second general requirement of the system is that it should employ non-technical (with regard to computers) language in dealing with the user. When developing the system, the design intent is to never lose sight of the user's needs: musicologists, while they deal in logical relations and decisions which utilize deductive reasoning, are not mathematicians. Often, they are not skilled in technical matters.

A third general requirement of the system is that it should provide the user with extensive storage capacities. Large research projects frequently reference hundreds of documents and music manuscripts. It is essential that the musicologist have a method of storing references to information which might be cited in the

final research product. To do this, the user of the system must be able to add, delete, update, or query the information stored.

3.3 Specific Requirements

This section addresses three requirements of the musicological tool: the user interface is addressed in section 3.3.1; the expert system in section 3.3.2; and the database management system in section 3.3.3.

3.3.1 User Interface

The user interface must be capable of providing the user with simple, easy-to-use menus, interpreting the commands entered by the user, and interfacing with the files provided by the graphics portion of the package. It must respond to user queries with appropriate responses, error messages, or additional user options. It must offer a simple, straightforward set of user instructions, requiring a minimal number of responses from the user.

3.3.2 Expert System

The expert system portion of the system must store high-level representations of the rules that professional

musicologists unconsciously use to interprete their materials. This knowledge will be customized for use by an individual researcher or musicological project to contain specific information for a particular research domain.

The musicological knowledge will be stored in a simple rule-based system. This knowledge base and its use as a research tool will be activated whenever an incipit is added to the data store or modified in any way.

While the basic tenets of the expert system are predefined by the research tool, additional rules may be added to the knowledge base in order to allow customization of the system. This feature is essential since the style considerations of the era which the musicological project is investigating may seriously alter the nature of the rules.

Furthermore, the expert system will allow user interaction: that is, the musicologist may contribute to the final decision of the expert system at break points in the decision process. In addition, the expert system will also communicate with the database system which serves as storage for the knowledge base and incipits.

3.3.3 Database Management System

The relational database management portion of the system must provide storage for the knowledge base of the expert system which consists of the complete set of rules constituting the general knowledge of the expert system. It must also store the domain-specific knowledge of the individual user's application.
Activation of the database management system will be initiated via three methods:

(1) a direct request by the user for retrieval, storage, or updating of information
(2) interaction with the graphical interface
(3) interaction with the expert system.

The database portion of the product must also allow the user to query or update information related to the composers and sources. Stored information will include such data as general information related to sources, probable composer, date of composition, location of manuscript or publication, and the musical incipits themselves.

4. DESIGN

4.1 Introduction

The primary issue addressed by the design chapter is that of integrating the expert system and database technology. This effort, combined with the research efforts of another Kansas State University project [Ha89] which addresses the integration of this design with a graphical interface, produces the research musicological tool. The design chapter concentrates on the incorporation of intelligence into the database processing environment so that the musicological data provided by the user can be intelligently and correctly stored in the musicological database. A second objective of this intelligence incorporation is that decision options can be offered to the user which assist in their research process. Section 4.2 provides a general discussion of the research musicological tool and presents the structural elements of the design effort. Section 4.3 addresses the specific design of the expert system. Section 4.4 addresses the design of the database of musicological information: incipits, composer information, manuscript and literary

references which may be utilized by the expert system. Section 4.5 discusses the interaction of the graphics interface with the expert system and the musicological database. This is followed by a brief summary of the system in section 4.6.

4.2 The Research Musicological Tool - An Overview

The purpose of the research musicological tool is to meet the needs of the independent research musicologist and the small music archives. The needs of this group include:

(1) a clear, concise, easy to use interface
(2) a means to store and manipulate textual data relating to composers and compositions
(3) a facility to enter, store, retrieve, and update musical incipits
(4) assistance in analyzing the data related to their project.

The research musicological tool addresses these needs by providing the following components: a menu-driven user interface which provides access to the musicological project database and the expert system; the musicological

database itself; the graphical interface; and the expert system with all its related maintenance facilities.

Section 4.2.1 discusses the basic structure of the tool. The structure of the music database facility is presented in Section 4.2.2. Finally, the structural division of the expert system maintenance subsystem is discussed in Section 4.2.3.

4.2.1 A Structural Overview

The user interface is the primary method whereby the user communicates with the system. It is, in essence, a scheduler for the system. Communication is maintained via simple-to-use menus and sub-menus. The initial menu offers the user access to the musicological database as well as the expert system maintenance facility.

The system is divided into three layers: (1) the high level menus; (2) the music database facility; and (3) the expert system maintenance facility. The first layer is represented in Illustration 4.1 which identifies the three primary menus.

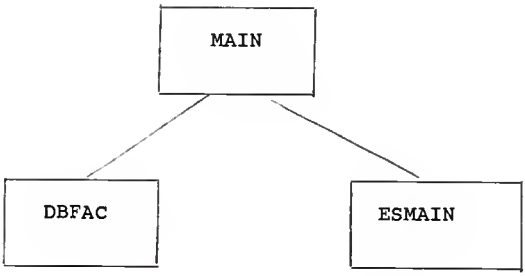


Illustration 4.1

MAIN contains the primary menu. It is the user's first introduction to the system, offering two options: DBFAC, which is the music database facility, and ESMAIN, the expert system maintenance facilities.

DBFAC is a secondary menu which provides the user with add, update, delete, and inquiry functions for the database of musical information.

ESMAIN serves as a secondary menu providing access to the rules employed in the expert system. The user is allowed to add rules, to activate or de-activate rules within the expert system, and to reorganize those rules within the

system as a means of customizing the system to meet the needs of the individual researcher.

4.2.2 DBFAC: The Music Database Facility

The music database facility is scheduled by the user interface whenever a user specifically requests to add, update, delete, or query entries in their file of musicological information. All incipit data is keyed by a system-assigned unique call number for simple retrieval. Source data is keyed by source title. Composer data is keyed by last name, first name, middle initial. Within DBFAC, both the graphics portion of the project and the expert system are key factors. If the intent of the user is to alter only textual data, neither the expert system utility nor the graphics portion of the project is invoked. Textual updates are the sole domain of the database facility. If, however, the user wishes to add, insert, update, or query an incipit, the graphics portion of the system is invoked in order to interface with the user. The add and update functions require that the expert system utility also be invoked as a means of checking for duplicate musical incipits. Thus, a researcher's direct request for a database function could

involve the graphics segment of the system, the expert system segment of the system, and the actual database facility as well.

The music database facility function includes four subdivisions: DBDEL, a delete function; DBADD, which provides an add function; DBUPD, an update function; and DBQUERY, the inquiry function. This relationship is shown in Illustration 4.2.

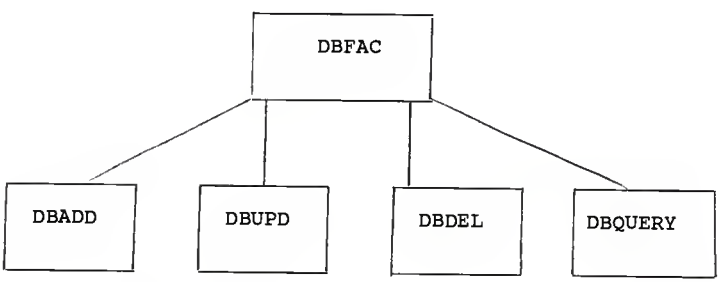


Illustration 4.2

4.2.2.1 DBADD

DBADD is a sub-menu which provides access to three separate functions: ADDINC allows the user to add an

incipit and its associated source and composer information; ADDCOMP allows the addition of an individual composer; and ADDSRC provides for the separate addition of information relating to sources. Illustration 4.3 represents this substructure.

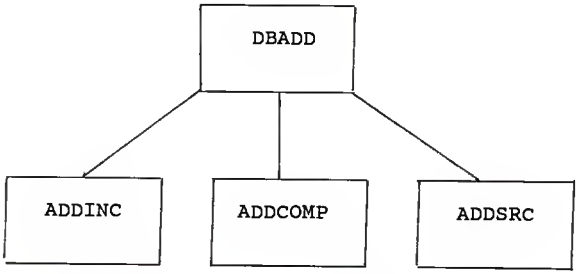


Illustration 4.3

Within this group, ADDINC is the heart of the system. The researcher's request to add an incipit schedules ADDINC. This function then schedules the graphics interface to allow entry of an incipit. Upon scheduling the interface, ADDINC builds an initial file to be used by the interface. This file consists of a function indicator ("A" for add); the proposed next system-

generated call number which will be assigned to the incipit if all expert system checks are sucessfully passed; space for the time signature; key signature; a major-minor indicator; and ten occurrences of associated notes, pitches, and accidentals.

Upon completion of the execution of the graphics interface module, the ADDINC module analyzes the incipit data via INFER, the inference engine. INFER itself schedules six additional modules to assist in the process:

(1) CHGNOTES changes all pitches to a common pattern, converting sharps or flats by adding or subtracting numeric values from the original numeric value of the pitch

(2) RHYTHM looks for rhythmic differences between the proposed incipit and all other incipits in the database, storing the result in a memory variable for later comparisons

(3) NOTES looks for actual pitch differences, storing this data in a memory variable for later comparisons

(4) TIME compares the time signature of the proposed addition to those of other incipits in the

database, storing the result in a memory variable

(5) TRANSP employs CHGNOTES, translating each of the incipits in the music database into a common form prior to examining them for possible transpositions

(6) Finally, RCHECK processes each of the rules stored in the knowledge base, allowing the researcher the ultimate decision concerning whether or not the incipit should be added to the database.

An analysis which produces no other similar incipits in the database allows the researcher to store this incipit. When similar incipits are found, the user is allowed to scroll through these incipits prior to making a decision to add the current incipit. At that point, both source and composer segments may be added to the database via the INCSADD and INCCADD segments. The flow of data through the total ADDINC process is shown in Illustration 4.4.

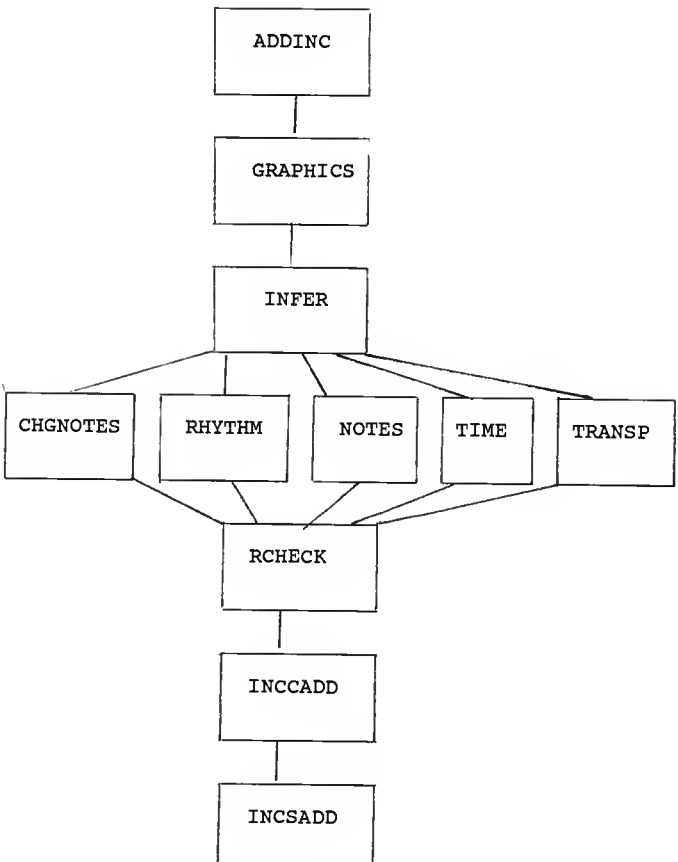


Illustration 4.4

ADDSRC and ADDCOMP add source and composer segments to the music database independent of the ADDINC all-inclusive model.

4.2.2.2 DBUPD

DBUPD, like DBADD, is a sub-menu program. Here the user is offered the opportunity to perform updates of incipit data, source data, or composer data. The structure of the database update function is shown in Illustration 4.5.

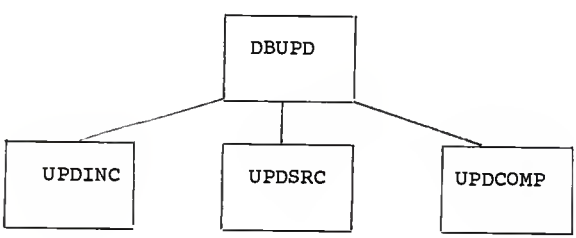


Illustration 4.5

Again, the incipit-processing module UPDINC interfaces with both the graphics program and the inference engine,

screening any changes which might result in a match on an existing incipit within the system.

4.2.2.3 DBDEL

DBDEL is also a sub-menu. All delete processing within the music database is done within this group. Similar in nature to DBUPD, this module allows delete processing for incipits, source records, and composer records. Unlike both ADDINC and UPDINC, DELINC schedules the graphics interface in retrieval mode as opposed to entry mode. During the delete process, the graphics interface allows the researcher to verify the incipit prior to deletion. As seen in Illustration 4.6, DBDEL provides access to the deletion of incipits and their associated sources and composers (DELINC), as well as the separate deletion of source segments (DELSRC) and composer segments (DELCOMP).

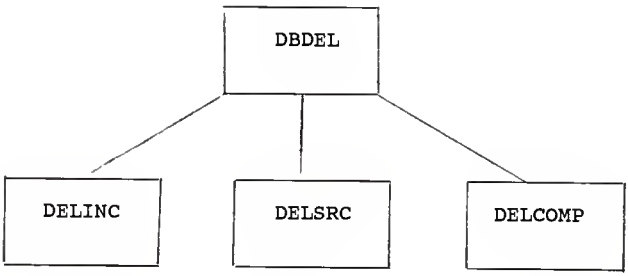


Illustration 4.6

4.2.2.4 DBQUERY

The final function within the music database facility is that of DBQUERY, a module which returns all the sources and composers associated with a given call number within the system.

4.2.3 ESMAIN: Expert System Maintenance

While both the database subsystem and the integrated expert system utility access the graphical interface, the second subsystem, that of expert system maintenance, has no ties to the graphical interface. Nor does it have strong ties to the musical incipit/composer database.

Rather, this subsystem controls its own database of musicological facts and rules. These form the heart of the decision process for the expert system. The basic rules of musicological research are predetermined by the knowledge engineering which must be accomplished prior to the implementation of the project; however, additional rules and fine tuning of the system are at the discretion of the user. In this way, the basic musicological tool can successfully be used by researchers working in a variety of musical eras. Without the ability to alter, add, delete, and restructure the importance of the various rules, the system would be inflexible and, consequently, less useful.

4.2.3.1 ESMAIN Components

Dynamic knowledge engineering must be part of a successful musicological tool: this is the role of ESMAIN, the expert system maintenance portion of the project. ESMAIN is essentially a subsystem which allows the user access to the knowledge base via five methods:

(1) ESADD provides the user with the capability to add rule types to the existing knowledge base

(2) ESRADD allows the addition of individual rules

within these rule types

(3) ESACT allows the user to activate or de-activate the various rule types -- a method of virtual insertion or deletion of the existing rules

(4) ESRACT allows this same activation/de-activation function to be performed on the rules within rule types

(5) ESREORG provides a sub-menu to rule reorganization functions.

This relationship is shown in Illustration 4.7.

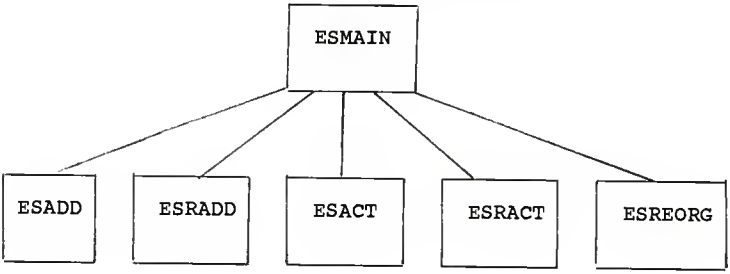


Illustration 4.7

4.2.3.2 ESREORG

ESREORG, represented in Illustration 4.8, is the only sub-menu within this group. It provides access to two

reorganizational features within the expert system: PRERULE1, which allows the researcher to designate the ordering of rule types within the knowledge base, and PRERULE2, which provides this same function for the rules within rule types.

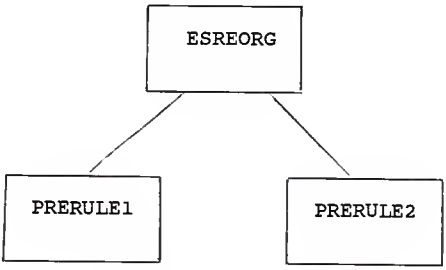


Illustration 4.8

4.3 The Expert System Structure

William R. Arnold and John S. Bowie, in their discussion of artificial intelligence, point out that while AI systems are diverse in their areas of expertise, all have (1) a knowledge base generally made up of rules, (2) an inference engine which performs the tasks that give the system intelligence, and (3) a maintenance engine which

is a program to update the knowledge base or inference engine or both [Ar86]. ESMAIN and its associated functions provide the maintenance engine for the research musicological tool. INFER and its related functions provide the inference engine. The knowledge base itself is provided by a database of rules.

4.3.1 Design of the Knowledge Base

The expert system database stores the knowledge base of musicological rules and musical facts gleaned from a close examination of the process whereby musicologists make intelligent decisions about the materials with which they work. These basic rules and facts may be supplemented by the individual musicologist using the system via the use of the expert system maintenance menus. The database is simple in design, reflecting the method employed by musicologists during the research process. Basically, the database consists of rules organized within rule types, each rule being tagged with the following attributes:

(1) an active-inactive tag which specifies whether the user wishes this rule type to be used in the decision process

(2) a sequence number tag which establishes priorities among the different types of rules

(3) a type indicator tag which indicates the basic category of rule

(4) a second active-inactive tag which allows individual rules within a type to be used

(5) a second sequence number tag which establishes the position of rules within each category

(6) the rule itself.

The combination of rule type, rule type sequence number, and rule sequence number allows each record to be unique. It also allows the researcher to specify that the same general rule be examined multiple times within the decision process, simply by allowing it to exist in multiple type categories. Suppose, for example, that a rule of type NOTES specified the range of note differences allowable between compositions before a match might be identified. This same rule might also exist within the type structure of COPY FAULTS and thus be reconsidered at the time that rule type was scheduled for examination.

Categories of rules might include:

(1) melodic configuration

(2) rhythmic configuration

(3) key considerations

(4) accidentals

(5) manuscript/print data

(6) composer data

(7) probable musical era data.

Obviously, the ideal system should allow new categories to be added by the user to customize the application.

Multiple rules exist in each area, all ordered according to their importance. The user has the option of accepting the standard knowledge base provided by the system. Optionally, the individual user may reorder the rules to suit his own application, insert additional rules, delete rules from the system, or build a new expert based on (1) his own rules, (2) a new ordering of existing rules, or (3) a mixture of existing and new rules. The goal is to provide the musicologist with total flexibility so that the tool can be tailored for each individual application.

4.3.2 Design of the Inference Engine

There are three general designs which typify expert system inference engines: (1) the forward-chaining, or data-driven, method in which a forward-chaining inference engine starts with some information and then tries to find an object that fits the information; (2) the backward-chaining, or object-driven, method whereby a backward-chaining inference engine starts with a hypothesis (an object) and requests information to confirm or deny it; and (3) the rule-value method in which a rule-value inference engine requests as its next piece of information the one which will remove the most uncertainty from the system [Sc87].

Vesonder, discussing rule-based programming in the UNIX system, states that today, most expert systems are built using a rule-based approach [Ve88]. A rule-based methodology, such as that chosen as the basis of the research musicological tool inference engine, is useful for two reasons:

(1) it allows the representation of units of knowledge in the form of rules

(2) because the chunks of knowledge are independent, they can be easily changed without altering the rest of the system.

Vesonder continues his discussion of the rule-based approach by saying that many expert systems have been built using rule-based tools because expertise stated in the form - "If this is the situation, then take this action" - is readily coded into the IF-THEN format of rules.

This format seems especially suited to the design of the research musicological tool. The inference engine designed for this project employs consecutive IF-THEN rule formats, grouping the rules in large chunks of knowledge which may actually be totally bypassed in the logic flow if that is the desire of the user.

4.4 The Database of Musicological Information

The database of musicological information, like that of the expert system rules, is simple in design.
It consists of a incipit segment keyed by a system-generated numeric call number linked to one or more composer segments keyed by composer name. The incipit

segments are also linked to one or more source segments which are uniquely keyed by a source title. An example of the source segment key might be Munich Bayerische Staatsbibliothek 5130. All three segments are linked to one another by the combination of call number, source title and composer name. Illustration 4.9 depicts the entity -relation structure of the musicological information database.

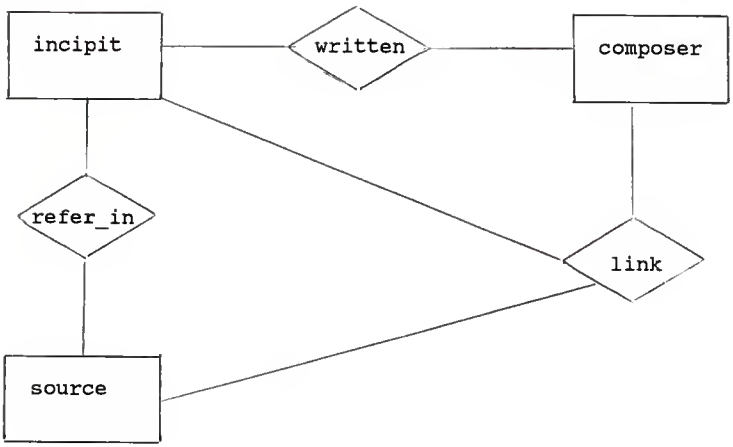


Illustration 4.9

4.5 The Graphical Interface and Expert System

The graphical interface provides the data which feeds the expert system. Whenever multiple similar versions of an incipit exist, the expert system presents these to the researcher via the graphical interface. The graphical interface also allows the user to enter new musical

incipits as part of the musicological database add and update capabilities of the system. The expert system then allows the user to pick and choose which incipits should remain in the working set when break points occur in the expert system logic and user interaction is required. Basically, the graphical interface accepts input from the user and converts this input into a record format which can be transferred to the expert system portion of the design. Once within the realm of the expert system utility, this same data is examined and tested for compliance with the criteria established by the knowledge base database. The graphical interface is never directly called by the user, yet it is an integral part of the total design.

4.6 System Summary

The research musicological tool consists of three primary elements, each of which could become a project in its own right: (1) the expert system; (2) the musical information database; and (3) the graphical interface. Soon after beginning the project, the enormity of the complete system became apparent. For this reason, the system is implemented with a "bare bones" approach. All

facets of the design are addressed; however, the expert system may not possess all the possible rules and rule combinations. Finally, the interaction between the graphical interface and the remainder of the system is addressed in a complete, yet abbreviated fashion. It is the intent of this project to offer the research musicologist a working model of what can be a useful addition to the world of musicological research.

5. IMPLEMENTATION

5.1 Introduction

This chapter addresses the implementation of the research musicological tool. It does not include references to the implementation of the graphical interface segment of the system: the design of this portion of the musicological tool was addressed in detail in another Kansas State University master's project [Ha89]. Those segments of the research musicological tool addressed in this chapter include: (1) the music information database; (2) the expert system with its associated inference engine, maintenance engine, and knowledge base; and (3) the interface between the graphical segment of the project and the database subsystem.

A brief overview of the implementation is given in Section 5.2. Section 5.3 addresses the implementation issues associated with the various databases employed in the research musicological tool. Section 5.4 provides a brief description of the implementation of the expert system itself. Section 5.5 addresses the interface to

the graphical segment of the project. Finally, Section 5.6 discusses the extent of the implementation.

5.2 Implementation Overview

One of the original requirements of the research musicological tool was that the system be implemented on a personal computer. In accordance with this requirement, the project was implemented on a AT&T 6300 personal computer using dBASE III+ Version 1.0. The graphical interface segment of the project [Ha89] used Turbo Pascal Version 5.0. Implementation of the project required two files (an input file and an output file) to pass data between the graphical interface and the database subsystem, a total of eight dBASE files, thirty-five dBASE III+ programs consisting of approximately 1880 lines of code and a Turbo Pascal program [Ha89].

5.3 Database Solutions

Databases were required for three separate parts of the total design effort: (1) the storage of information related to the musical incipits, sources, and composers; (2) the storage of the rules which form the knowledge

base for the expert system; and (3) the intermediate storage of data passed as a file between the graphical interface and the database portion of the project.

The storage of information related to music incipits, sources, and composers required six files. Three of these were used for actual storage of information input by the researcher using the tool. An additional three were used to establish linkage between the data provided by the user. The tool established six separate database work areas with files active in each. Implementation of the research tool did not address null values in the key fields. The files employed for the implementation of the database of musical information are listed in Table 5.1.

TABLE 5.1

MUSICAL INFORMATION DATABASE: STRUCTURES

Database	Field Name	Type	Width
MUSIC	Call_num	Numeric	5
	Key	Character	2
	Title	Character	80
	MM	Character	1
	Time1	Numeric	1
	Time2	Numeric	1
	Inc_d1	Numeric	5
	Inc_p1	Numeric	2
	Inc_a1	Character	1
	Inc_d2	Numeric	5
	Inc_p2	Numeric	2

	Inc_a2	Character	1
	Inc_d3	Numeric	5
	Inc_p3	Numeric	2
	Inc_a3	Character	1
	Inc_d4	Numeric	5
	Inc_p4	Numeric	2
	Inc_a4	Character	1
	Inc_d5	Numeric	5
	Inc_p5	Numeric	2
	Inc_a5	Character	1
	Inc_d6	Numeric	5
	Inc_p6	Numeric	2
	Inc_a6	Character	1
	Inc_d8	Numeric	5
	Inc_p8	Numeric	2
	Inc_a8	Character	1
	Inc_d9	Numeric	5
	Inc_p9	Numeric	2
	Inc_a9	Character	1
	Inc_d10	Numeric	5
	Inc_p10	Numeric	2
	Inc_a10	Character	1
COMPOSER	Comp_bdate	Character	4
	Comp_ddate	Character	4
	Comp_loc	Character	40
	Comp_lname	Character	20
	Comp_fname	Character	10
	Comp -init	Character	1
SOURCE	Src_type	Character	40
	Src_title	Character	40
	Src_date	Character	4
WRITTEN	Call_num	Numeric	5
	Comp_lname	Character	20
	Comp_fname	Character	10
	Comp_init	Character	1
REFER_IN	Call_num	Numeric	5
	Comp_lname	Character	20
	Comp_fname	Character	10
	Comp_init	Character	1
LINK	Call_num	Numeric	5
	Comp_lname	Character	20
	Comp_fname	Character	10
	Comp_init	Character	1

Src_title	Character	40

The index structures associated with the database are listed in Table 5.2.

TABLE 5.2

MUSICAL INFORMATION DATABASE: INDEX STRUCTURES

Database	**Index Name**	**Database Fields**
MUSIC	Call_num	Call_num
SOURCE	Sources	Src-title
COMPOSER	C_Lname	Comp_lname + Comp_fname + Comp_init
WRITTEN	Callw	Call_num
	Cnamew	Comp_lname + Comp_fname + Comp_init
REFER-IN	Callr	Call_num
	Srcr	Src_title
LINK	Calll	Call _num
	Cnamel	Comp_lname + Comp_fname + Comp_init
	Srcl	Src_title

The expert system required a single database. The structure of this database and the index associated with it are shown in Table 5.3.

TABLE 5.3

EXPERT SYSTEM DATABASE AND INDEX: STRUCTURES

Database	Field Name	Type	Width
RULE	Type	Character	20
	Seq_num1	Numeric	3
	Seq_num2	Numeric	3
	Act_flag1	Logical	1
	Act_flag2	Logical	1
	Field1	Character	39
	Oper	Character	2
	Field	Character	39
Database	**Index Name**	**Database fields**	
RULE	Num1	Seq_num1	
	Num2	Seq_num2	
	Types	type	

Finally, the temporary database which was required by dBASE III+ as a means of communicating with a file is shown in Table 5.4 with its associated index structure.

TABLE 5.4

TEMPORARY DATABASE AND INDEX: STRUCTURES

Database	Field Name	Type	Width
GRAPHF	Act	Character	1
	Call	Numeric	5
	Key	Character	2
	MM	Character	1
	Time1	Numeric	1
	Time2	Numeric	1
	Inc_d1	Numeric	5
	Inc_p1	Numeric	2

Inc_a1	Character	1
Inc_d21	Numeric	5
Inc_p21	Numeric	2
Inc_a21	Character	1
Inc_d31	Numeric	5
Inc_p31	Numeric	2
Inc_a31	Character	1
Inc_d41	Numeric	5
Inc_p41	Numeric	2
Inc_a41	Character	1
Inc_d51	Numeric	5
Inc_p51	Numeric	2
Inc_a51	Character	1
Inc_d61	Numeric	5
Inc_p61	Numeric	2
Inc_a61	Character	1
Inc_d71	Numeric	5
Inc_p71	Numeric	2
Inc_a71	Character	1
Inc_d81	Numeric	5
Inc_p81	Numeric	2
Inc_a81	Character	1
Inc_d91	Numeric	5
Inc_p91	Numeric	2
Inc_a91	Character	1
Inc_d10	Numeric	5
Inc_p10	Numeric	2
Inc_a10	Character	1

Database	**Index Name**	**Database Fields**
GRAPHF	Callg	call

A study of the tables of database structures reveals one of the problems inherent in dBASE III+: there are no arrays. This, coupled with the 254-characters-per-line restriction imposed by dBASE III+, proved to be a true impediment since it was necessary to pass long lists of note duration and pitch parameters between programs. In addition to the lack of array structures, dBASE III+ has

a restriction regarding the number of files which can be open at one time. Since each program and database structure or index structure is considered to be a file, this was a particular problem. In order to circumvent the files issue, all the programs were placed within one large procedure. Thus, the restriction that only one procedure be active required all parameter passing to occur program to program rather than via the use of additional procedures.

5.4 Expert System Solutions

A large part of any design involving an expert system is knowledge engineering. In order to devise a system which can do the work of the expert, care must be taken to thoroughly understand the steps taken by a musicologist during the research process. Several years experience spent by the author working on a National Endowment for the Humanities research project simplified this process.

One of the more interesting problems associated with the project was that of devising a method whereby the researcher could actually influence the structure of the expert system, dynamically altering the rules within the

knowledge base. Within dBASE III+, a feature exists which allows field names to be used as macros. This instigated the division of the rule itself into three segments: field1, operator, and field2. These were then treated as macros. By limiting the user to selected field names and operators, the system actually provides the user with the capability to set acceptable limits of pitch variations, rhythmic differences, and time signature variations.

5.5 Graphical Interface Solutions

At the outset of the project, the problem of communicating between the database subsystem and the graphical interface seemed insurmountable. However, the ultimate solution was surprisingly simple: it involved employing two files and a temporary dBASE database which was required to match the structure of these two files exactly and the execution of a DOS command within a dBASE program. Whenever an add or update of an incipit is requested by the user, the database subsystem builds a temporary GRAPHF database record with information from the permanent MUSIC database. This database record is used to generate a file. At that point, a DOS command is

issued within a dBASE program to execute the Pascal-based graphical interface. Upon completion of execution, the dBASE program regains control, reading the second file which is generated by the graphical interface and storing it in the temporary GRAPHF database for comparison to existing incipits stored in the MUSIC database.

5.6 Extent

The intent of the implementation was to provide a working tool for the researcher in musicology. In order to do this, all features of the project were implemented with two exceptions:

(1) The TRANSP module within the expert system was not implemented. The logic necessary for this was also used as the basis for the NOTES module. This, implementation of TRANSP was deemed non-essential.

(2) The SCROLL capability within the ADDINC module was not implemented. This feature is useful, but not essential. The design intent of SCROLL was to allow the user to view multiple similar versions of incipits residing on the database prior to choosing to override the expert's decision to not add another similar incipit.

6. Conclusions and Extensions

6.1 Introduction

The motivation of this project was to provide an automated research tool for musicologists and small music archives and to explore the usefulness of expert systems as a potential research tool for music historians.

6.2 Project Conclusions

In the course of completing this project, the following results were achieved:

* Analyzed the problem of providing a research tool for the computer-novice professional musicologist.

* Designed a simple-to_use interactive system employing expert system functions to assist in incipit matching and a database management system to store musicological research information.

* Implemented the research musicological tool for the personal computer using dBASE III+ and a rule-based inference engine.

* Achieved a successful integration of dBASE III+, expert system functions, and a graphical interface to produce a useful tool for scholars in the field of historical musicology.

6.3 Future Enhancements

The research musicological tool can be viewed as a beginning point -- a first step toward mechanized aids for musicologists. In light of that, there are several topics within the scope of the project which would benefit from additional research. The following are recommended areas for further research and development:

* The addition of free form notes to the database subsystem would be particularly beneficial to researchers in any historical discipline.

* The porting of the system to FoxBASE+ or dBASE IV would provide flexibility for the

development of a more functionally-rich product, allowing the features of dBASE without restrictions on array processing and files.

* The design of the current project did not include report capabilities, yet this is an important tool for the research musicologist. Printed lists of incipits and associated composers/sources would be especially helpful during the preparation of articles and papers based on data stored in the research musicological tool.

* A help function which would allow the user to enter an incipit, then view all similar incipits within the system should be implemented to increase the usefulness of the tool.

REFERENCES

[Ap72] Apel, Willi. Harvard Dictionary of Music. Cambridge, Massachusetts: Harvard University Press, 1972.

[Ar86] Arnold, William R. and John S. Bowie. Artificial Intelligence: A Personal, Commonsense Journey. Englewood Cliffs, New Jersey: Prentice-Hall, Inc., 1986.

[Ba87] Bartle, Barton K. Computer Software in Music and Music Education. Metuchen, New Jersey: Scarecrow Press, 1987.

[Br86] Bratko, Ivan. Prolog Programming for Artificial Intelligence. Workingham, England: Addison-Wesley Publishing Company, 1986.

[Br88] Brachman, Ronald J. "The Basics of Knowledge Representation and Reasoning." AT&T Technical Journal, 67, No. 1 (1988), pp. 7-24.

[Ch84] Charnasse, Helene. "Les bases de donnees en musicologie." Fontes Artis Musicae, 31, No. 3 (1984), pp. 153-159.

[Ch87] Chabris, Christopher F. Artificial Intelligence and Turbo Pascal. Homewood, Illinois: Dow Jones-Irwin, 1987.

[Da87] Date, C. J. An Introduction to Database Systems. Reading, Massachusetts: Addison-Wesley Publishing Company, 1987.

[Dr84] Drummond, Philip J. "Developing Standards for Musicological Databases." Fontes Artis Musicae, 31, No. 4 (1984), pp. 172-176.

[Fo86] Forsyth, Richard. "The Anatomy of Expert Systems." In Artificial Intelligence: Principles and Application. Edited by Masoud Yazdani. London: Chapman and Hall, 1986.

[Go84] Gooch, Bryan N. S. "Catalogues and Computers or Bibliography and the Beast." Fontes Artis Musicae, 31, No. 1 (1984), pp 38-41.

[Ha89] Harmon, Karen LuAnne. "A Graphical Interface for Musicological Research Tools". Master's Report, Kansas State University, 1989.

[He86] Hewlett, Walter and Eleanor Selfridge-Field. Directory of Computer-Assisted Research in Musicology. Menlo Park, California: Center for Computer-Assisted Research in the Humanities, 1986.

[Ho88] Hofstetter, Fred Thomas. Computer Literacy for Musicians. Englewood Cliffs, New Jersey: Prentice-Hall, 1988.

[Hu84] Hultberg, Warren E. "Data Bases for the Study of Relationships among Spanish Music Sources of the 16th-17th Centuries." Fontes Artis Musicae, 31, No. 3 (1984), pp 162-167.

[Le85] Lehner, Paul E. and Stephen W. Barth. "Expert Systems on Microcomputers." In Applications in Artificial Intelligence. Princeton, N.J.: Petrocelli Books, 1985.

[Le86] Levine, Robert I., Diane E. Drang, and Barry Edelson. A Comprehensive Guide to AI and Expert Systems. New York: McGraw-Hill Book Company, 1986.

[Li84] Lincoln, Harry B. "A Description of the Database in Italian Secular Polyphony Held at SUNY-Binghamton, New York." Fontes Artis Musicae, 31, No. 3 (1984), pp 159-162.

[Mo84] Morehen, John. "Thematic Cataloging by Computer." Fontes Artis Musicae, 31, No. 1 (1984), pp. 32-38.

[Pa88] Parker, Barbara S. "Incorporating Expert System Technology into a Professional Genealogical Information System." Master's Thesis, Kansas State University, 1988.

[Pe89] Pedersen, Ken. Expert Systems Programming: Practical Technique for Rule-Based Systems. New York: John Wiley & Sons, 1989.

[Ra89] Ranada, David. "Music from Machines - Part I: Harmonizing Bach-style Chorales: IBM does it all." Musical America, March (1989), pp. 95-96.

[Sc87] Schildt, Herbert. Artificial Intelligence Using C. Berkeley, California: Osborne McGraw-Hill, 1987.

[Sk88] Skinner, Robert. "Microcomputers in the Music Library." Notes: Quarterly Journal of the Music Library Association, 45, No. 1 (1988), pp. 7-14.

[St81] Steelman, Robert. Catalog of the Lititz Congregation Collection. Chapel Hill, North Carolina: University of North Carolina Press, 1981.

[Ve88] Vesonder, Gregg T. "Rule-Based Programming in the UNIX System.", AT&T Technical Journal, 67, No. 1 (1988), pp. 69-80.

[Wi84] Winston, Patrick Henry. Artificial Intelligence. Reading, Massachusetts: Addison-Wesley Publishing Company, 1984.

1. OVERVIEW 1
1.1 Introduction 1
1.2 Literary Review 4
1.2.1 Database Design 4
1.2.2 Expert Systems 6
1.2.3 Musicological Resources 9
1.3 Summary 13
2. THE PROCESS OF MUSICOLOGICAL RESEARCH 14
2.1 Musicology as a Discipline 14
2.2 The Typical Musicologist 15
2.3 The Research Project 16
2.4 Research Procedures 17
2.5 Research Product and the Proposed Research Tool 19
3. REQUIREMENTS 21
3.1 Introduction 21
3.2 General Requirements 23
3.3 Specific Requirements 25
3.3.1 User Interface 25
3.3.2 Expert System 25
3.3.3 Database Management System 27
4. DESIGN 28
4.1 Introduction 28
4.2 The Research Musicological Tool - An Overview 29
4.2.1 A Structural Overview 30
4.2.2 DBFAC
The Music Database Facility 32
4.2.2.1 DBADD 33
4.2.2.2 DBUPD 38
4.2.2.3 DBDEL 39
4.2.2.4 DBQUERY 40
4.2.3 ESMAIN
Expert System Maintenance 40
4.2.3.1 ESMAIN Components 41
4.2.3.2 ESREORG 42
4.3 The Expert System Structure 43
4.3.1 Design of the Knowledge Base 44
4.3.2 Design of the Inference Engine 47
4.4 The Database of Musicological Information 48
4.5 The Graphical Interface and Expert System 50
4.6 System Summary 51
5. IMPLEMENTATION 53
5.1 Introduction 53
5.2 Implementation Overview 54
5.3 Database Solutions 54
5.4 Expert System Solutions 60
5.5 Graphical Interface Solutions 61
5.6 Extent 62
6. Conclusions and Extensions 63
6.1 Introduction 63
6.2 Project Conclusions 63

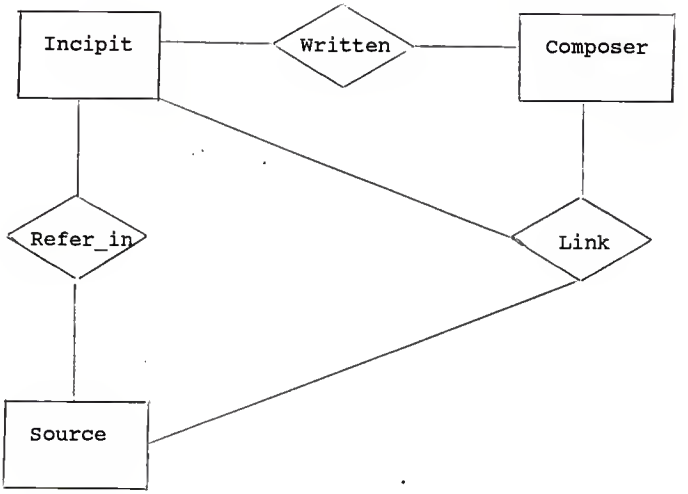
LIST OF ILLUSTRATIONS

Illustration 2.1 Examples of Musicological Project....18
Illustration 3.1 System Design.....23
Illustration 4.1 High Level Menus.....31
Illustration 4.2 Database Facility Structure.....33
Illustration 4.3 DBADD Subsystem.....34
Illustration 4.4 Expert System Data Flow.....37
Illustration 4.5 DBUPD Subsystem.....38
Illustration 4.6 DBDEL Subsystem.....40
Illustration 4.7 Expert System Maintenance Structure..42
Illustration 4.8 ESREORG Subsystem.....43
Illustration 4.9 E-R Structure of Musical Information.50

LIST OF TABLES

Table 5.1 Musical Information Database Structures.....55
Table 5.2 Musical Information Database Index Structures.....57
Table 5.3 Expert System Database and Index: Structures.....58
Table 5.4 Temporary Database and Index: Structure.....58

RELATIONAL SCHEMAS MUSIC DATABASE




```
Incipit (Key, Call_num, Title, MM, Time1, Time2,
     Inc_d1, Inc_d2, Inc_d3, Inc_d4, Inc_d5, Inc_d6,
     Inc_d7, Inc_d8, Inc_d9, Inc_d10, Inc_p1, Inc_p2,
     Inc_p3, Inc_p4, Inc_p5, Inc_p6, Inc_p7, Inc_p8,
     Inc_p9, Inc_p10, Inc_a1, Inc_a2, Inc_a3, Inc_a4,
     Inc_a5, Inc_a6, Inc_a7, Inc_a8, Inc_a9, Inc_a10)
     key: Call_num

Source (Src_type, Src_title, Src_date)
     key: Src_title

Composer (Comp_bdate, Comp_ddate, Comp_loc, Comp_lname,
          Comp_fname, Comp_init)
     key: (Comp_lname, Comp_fname, Comp_init)

Written_by (Call_num, Comp_lname, Comp_fname, Comp_init)
     key: (Call_num, Comp_lname, Comp_fname, Comp_init)
```

Refer_in (Call_num, Src_title)
key: (Call_num, Src_title)

Link (Call_num, Src_title, Comp_lname, Comp_fname, Comp_init)
key: (Call_num, Src_title, Comp_lname, Comp_fname, Comp_init)

DATA DICTIONARY

Data Name	Inc_d1
Aliases	None
Data type	Numeric
Format	5 numerics
Range	10000 = whole note 07500 = dotted half note 05000 = half note 03750 = dotted quarter note 02500 = quarter note 01875 = dotted eighth note 01250 = eighth note 00625 = sixteenth note
Responsibility	
Security	
Availability	
Frequency	
Dependencies	
Comments	Originally received from graphical interface

Data Name	Inc_d2
Aliases	None
Data type	Numeric
Format	5 numerics
Range	10000 = whole note 07500 = dotted half note 05000 = half note 03750 = dotted quarter note 02500 = quarter note

	01875 = dotted eighth note 01250 = eighth note 00625 = sixteenth note
Responsibility	
Security	
Availability	
Frequency	
Dependencies	
Comments	Originally received from graphical interface

Data Name	Inc_d3
Aliases	None
Data type	Numeric
Format	5 numerics
Range	10000 = whole note 07500 = dotted half note 05000 = half note 03750 = dotted quarter note 02500 = quarter note 01875 = dotted eighth note 01250 = eighth note 00625 = sixteenth note
Responsibility	
Security	
Availability	
Frequency	
Dependencies	
Comments	Originally received from graphical interface

Data Name	Inc_d4
Aliases	None
Data type	Numeric
Format	5 numerics
Range	10000 = whole note 07500 = dotted half note 05000 = half note 03750 = dotted quarter note 02500 = quarter note 01875 = dotted eighth note

01250 = eighth note
00625 = sixteenth note

Responsibility
Security
Availability
Frequency
Dependencies
Comments — Originally received from graphical interface

Data Name — Inc_d5
Aliases — None
Data type — Numeric
Format — 5 numerics
Range —
10000 = whole note
07500 = dotted half note
05000 = half note
03750 = dotted quarter note
02500 = quarter note
01875 = dotted eighth note
01250 = eighth note
00625 = sixteenth note

Responsibility
Security
Availability
Frequency
Dependencies
Comments — Originally received from graphical interface

Data Name — Inc_d6
Aliases — None
Data type — Numeric
Format — 5 numerics
Range —
10000 = whole note
07500 = dotted half note
05000 = half note
03750 = dotted quarter note
02500 = quarter note
01875 = dotted eighth note

	01250 = eighth note 00625 = sixteenth note
Responsibility	
Security	
Availability	
Frequency	
Dependencies	
Comments	Originally received from graphical interface

Data Name	Inc_d7
Aliases	None
Data type	Numeric
Format	5 numerics
Range	10000 = whole note 07500 = dotted half note 05000 = half note 03750 = dotted quarter note 02500 = quarter note 01875 = dotted eighth note 01250 = eighth note 00625 = sixteenth note
Responsibility	
Security	
Availability	
Frequency	
Dependencies	
Comments	Originally received from graphical interface

Data Name	Inc_d8
Aliases	None
Data type	Numeric
Format	5 numerics
Range	10000 = whole note 07500 = dotted half note 05000 = half note 03750 = dotted quarter note 02500 = quarter note 01875 = dotted eighth note

	01250 = eighth note 00625 = sixteenth note
Responsibility	
Security	
Availability	
Frequency	
Dependencies	
Comments	Originally received from graphical interface

Data Name	Inc_d9
Aliases	None
Data type	Numeric
Format	5 numerics
Range	10000 = whole note 07500 = dotted half note 05000 = half note 03750 = dotted quarter note 02500 = quarter note 01875 = dotted eighth note 01250 = eighth note 00625 = sixteenth note
Responsibility	
Security	
Availability	
Frequency	
Dependencies	
Comments	Originally received from graphical interface

Data Name	Inc_d10
Aliases	None
Data type	Numeric
Format	5 numerics
Range	10000 = whole note 07500 = dotted half note 05000 = half note 03750 = dotted quarter note

02500 = quarter note
01875 = dotted eighth note
01250 = eighth note
00625 = sixteenth note

Responsibility
Security
Availability
Frequency
Dependencies
Comments: Originally received from graphical interface

Data Name: Inc_p1
Aliases: None
Data type: Numeric
Format: 2 numerics
Range:
00 = rest
90 = blank pitch
99 = bar
02 - 22 = notes, in even increments

Responsibility
Security
Availability
Frequency
Dependencies
Comments: Originally received from graphical interface via temporary database

Data Name: Inc_p2
Aliases: None
Data type: Numeric
Format: 2 numerics
Range:
00 = rest
90 = blank pitch
99 = bar
02 - 22 = notes, in even increments

Responsibility
Security

Availability	
Frequency	
Dependencies	
Comments	Originally received from graphical interface via temporary database

Data Name	Inc_p3
Aliases	None
Data type	Numeric
Format	2 numerics
Range	00 = rest 90 = blank pitch 99 = bar 02 - 22 = notes, in even increments
Responsibility	
Security	
Availability	
Frequency	
Dependencies	
Comments	Originally received from graphical interface via temporary database

Data Name	Inc_p4
Aliases	None
Data type	Numeric
Format	2 numerics
Range	00 = rest 90 = blank pitch 99 = bar 02 - 22 = notes, in even increments
Responsibility	
Security	
Availability	
Frequency	
Dependencies	
Comments	Originally received from graphical interface via temporary database

Data Name	Inc_p5
Aliases	None
Data type	Numeric
Format	2 numerics
Range	00 = rest 90 = blank pitch 99 = bar 02 - 22 = notes, in even increments
Responsibility	
Security	
Availability	
Frequency	
Dependencies	
Comments	Originally received from graphical interface via temporary database

Data Name	Inc_p6
Aliases	None
Data type	Numeric
Format	2 numerics
Range	00 = rest 90 = blank pitch 99 = bar 02 - 22 = notes, in even increments
Responsibility	
Security	
Availability	
Frequency	
Dependencies	
Comments	Originally received from graphical interface via temporary database

Data Name	Inc_p7
Aliases	None
Data type	Numeric
Format	2 numerics
Range	00 = rest 90 = blank pitch 99 = bar

02 - 22 = notes, in even increments

Responsibility
Security
Availability
Frequency
Dependencies
Comments: Originally received from graphical interface via temporary database

Data Name: Inc_p8
Aliases: None
Data type: Numeric
Format: 2 numerics
Range:
00 = rest
90 = blank pitch
99 = bar
02 - 22 = notes, in even increments

Responsibility
Security
Availability
Frequency
Dependencies
Comments: Originally received from graphical interface via temporary database

Data Name: Inc_p9
Aliases: None
Data type: Numeric
Format: 2 numerics
Range:
00 = rest
90 = blank pitch
99 = bar
02 - 22 = notes, in even increments

Responsibility
Security
Availability
Frequency
Dependencies

Comments	Originally received from graphical interface via temporary database

Data Name	Inc_p10
Aliases	None
Data type	Numeric
Format	2 numerics
Range	00 = rest 90 = blank pitch 99 = bar 02 - 22 = notes, in even increments
Responsibility	
Security	
Availability	
Frequency	
Dependencies	
Comments	Originally received from graphical interface via temporary database

Data Name	Inc_a1
Aliases	None
Data type	Character
Format	1 character
Range	+ = sharp - = flat 0 = natural
Responsibility	
Security	
Availability	
Frequency	
Dependencies	
Comments	Originally received from graphical interface via temporary database

Data Name	Inc_a2
Aliases	None

Data type	Character
Format	1 character
Range	+ = sharp - = flat 0 = natural
Responsibility	
Security	
Availability	
Frequency	
Dependencies	
Comments	Originally received from graphical interface via temporary database

Data Name	Inc_a3
Aliases	None
Data type	Character
Format	1 character
Range	+ = sharp - = flat 0 = natural
Responsibility	
Security	
Availability	
Frequency	
Dependencies	
Comments	Originally received from graphical interface via temporary database

Data Name	Inc_a4
Aliases	None
Data type	Character
Format	1 character
Range	+ = sharp - = flat 0 = natural
Responsibility	
Security	
Availability	
Frequency	

Dependencies	
Comments	Originally received from graphical interface via temporary database

Data Name	Inc_a5
Aliases	None
Data type	Character
Format	1 character
Range	+ = sharp - = flat 0 = natural
Responsibility	
Security	
Availability	
Frequency	
Dependencies	
Comments	Originally received from graphical interface via temporary database

Data Name	Inc_a6
Aliases	None
Data type	Character
Format	1 character
Range	+ = sharp - = flat 0 = natural
Responsibility	
Security	
Availability	
Frequency	
Dependencies	
Comments	Originally received from graphical interface via temporary database

Data Name	Inc_a7

Aliases	None
Data type	Character
Format	1 character
Range	+ = sharp - = flat 0 = natural
Responsibility	
Security	
Availability	
Frequency	
Dependencies	
Comments	Originally received from graphical interface via temporary database

Data Name	Inc_a8
Aliases	None
Data type	Character
Format	1 character
Range	+ = sharp - = flat 0 = natural
Responsibility	
Security	
Availability	
Frequency	
Dependencies	
Comments	Originally received from graphical interface via temporary database

Data Name	Inc_a9
Aliases	None
Data type	Character
Format	1 character
Range	+ = sharp - = flat 0 = natural
Responsibility	
Security	
Availability	
Frequency	

Dependencies	
Comments	Originally received from graphical interface via temporary database

Data Name	Inc_a10
Aliases	None
Data type	Character
Format	1 character
Range	+ = sharp - = flat 0 = natural
Responsibility	
Security	
Availability	
Frequency	
Dependencies	
Comments	Originally received from graphical interface via temporary database

Data Name	Key
Aliases	
Data type	Character
Format	2 characters
Range	First character: A - G Second character: #, b, blank
Responsibility	
Security	
Availability	
Frequency	
Dependencies	
Comments	

Data Name	Call_num
Aliases	
Data type	Numeric

Format	5 numerics
Range	00000 through 99999
Responsibility	System-generated
Security	
Availability	
Frequency	
Dependencies	
Comments	Generated by the system for means of having unique number system

Data Name	Time1
Aliases	
Data type	Numeric
Format	1 digit
Range	1 - 9
Responsibility	Created by graphical interface
Security	
Availability	
Frequency	
Dependencies	
Comments	Denotes the first number in a time signature

Data Name	Time2
Aliases	
Data type	Numeric
Format	1 digit
Range	1 - 9
Responsibility	Created by graphical interface
Security	
Availability	
Frequency	
Dependencies	
Comments	Denotes the second number in a time signature

Data Name	MM
Aliases	

Data type	Character
Format	2 characters
Range	blank = major m = minor
Responsibility	
Security	
Availability	
Frequency	
Dependencies	
Comments	Indicates major or minor key

Data Name	Title
Aliases	
Data type	Character
Format	80 characters
Range	
Responsibility	
Security	
Availability	
Frequency	
Dependencies	
Comments	Title of the composition

Data Name	Src_type
Aliases	
Data type	Character
Format	40 characters
Range	
Responsibility	
Security	
Availability	
Frequency	
Dependencies	
Comments	Indicates type of source, whether a manuscript (copied by hand) or printed version of the music

Data Name	Src_title

Aliases	
Data type	Character
Format	40 characters
Range	
Responsibility	
Security	
Availability	
Frequency	
Dependencies	
Comments	Title of the manuscript or print

Data Name	Src_date
Aliases	
Data type	Character
Format	yyyy
Range	0000 <= yyyy <= (current year)
Responsibility	
Security	
Availability	
Frequency	
Dependencies	
Comments	Year of manuscript or print origin

Data Name	Comp_bdate
Aliases	
Data type	Character
Format	yyyy
Range	yyyy <= (current year)
Responsibility	
Security	
Availability	
Frequency	
Dependencies	
Comments	Birth year of composer

Data Name	Comp_ddate
Aliases	
Data type	Character
Format	yyyy
Range	comp_bdate <= yyyy <= (current year)
Responsibility	
Security	
Availability	
Frequency	
Dependencies	
Comments	Death year of composer

Data Name	Comp_loc
Aliases	
Data type	Character
Format	40 characters
Range	
Responsibility	
Security	
Availability	
Frequency	
Dependencies	
Comments	Geographical location of composer

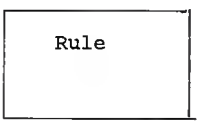
Data Name	Comp_lname
Aliases	
Data type	Character
Format	20 characters
Range	
Responsibility	
Security	
Availability	
Frequency	
Dependencies	Combination of last name, first name, and initial must be unique

	key
Comments	Composer's last name

Data Name	Comp_fname
Aliases	
Data type	Character
Format	10 characters
Range	
Responsibility	
Security	
Availability	
Frequency	
Dependencies	Combination of last name, first name, and initial must be unique key
Comments	First name of composer

Data Name	Comp_init
Aliases	
Data type	Character
Format	1 character
Range	
Responsibility	
Security	
Availability	
Frequency	
Dependencies	Combination of last name, first name, and initial must be unique key
Comments	Middle initial of composer

RELATIONAL SCHEMAS RULE DATABASE



Rule (Act_flag1, Act_flag2, Seq_num1, Seq_num2, Type,
Field1, Oper, Field2)
key: (Type, Seq_num1, Seq_num2)

DATA DICTIONARY

Data Name	Act_flag1
Aliases	None
Data type	Logical
Format	1 character
Range	"Y" for active, "N" for inactive
Responsibility	
Security	
Availability	
Frequency	
Dependencies	
Comments	Designates whether the entire family of rules of that "type" is active (that is, to be used in the decision process) or not

Data Name	Act_flag2
Aliases	
Data type	Logical
Format	1 character
Range	"Y" for active, "N" for inactive
Responsibility	
Security	
Availability	
Frequency	
Dependencies	
Comments	Designates whether a rule of this specific type is active or active

Data Name	Seq_num1
Aliases	
Data type	Numeric
Format	3 digits
Range	001 through 999
Responsibility	
Security	
Availability	
Frequency	
Dependencies	
Comments	Order of importance for the rule types. All rules with a sequence number of 3 would be examined prior to rules with a sequence number of 4.

Data Name	Seq_num2
Aliases	
Data type	numeric
Format	3 digits
Range	001 through 999
Responsibility	
Security	
Availability	
Frequency	
Dependencies	
Comments	Order of importance of rules within a single rule type

Data Name	Type
Aliases	
Data type	Character
Format	20 characters
Range	
Responsibility	
Security	
Availability	
Frequency	
Dependencies	
Comments	Subject matter with which each rule deals

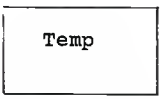
Data Name	Field1
Aliases	
Data type	Character
Format	39 characters
Range	TYPEDIF, NOTEDIF, TIME, RHYDIF, all numbers
Responsibility	
Security	
Availability	
Frequency	
Dependencies	Must be used as first macro in group of three: Field1 Oper Field2
Comments	These will be converted to macros in DBase III+, then used in IF-THEN statements (IF Macro1 Macro2 Macro3 THEN format)

Data Name	Oper
Aliases	
Data type	Character
Format	02 characters
Range	< , > , =, <>, <=, >=
Responsibility	
Security	
Availability	
Frequency	
Dependencies	Must be used as second macro in

	group of three: Field1 Oper Field2
Comments	These will be converted to macros in DBase III+, then used in IF-THEN statements (IF Macro1 Macro2 Macro3 THEN format)

Data Name	Field2
Aliases	
Data type	Character
Format	39 characters
Range	TYPEDIF, NOTEDIF, TIME, RHYDIF, all numbers
Responsibility	
Security	
Availability	
Frequency	
Dependencies	Must be used as third macro in a group of three: Field1 Oper Field3
Comments	These will be converted to macros in DBase III+, then used in IF-THEN statements (IF Macro1 Macro2 Macro3 THEN format)

RELATIONAL SCHEMAS GRAPHF DATABASE




```
Temp (Act, Key, Call, MM, Time1, Time2, Inc_d1, Inc_p1,
        Inc_a1, Inc_d2, Inc_p2, Inc_a2, Inc_d3, Inc_p3,
        Inc_a3, Inc_d4, Inc_p4, Inc_a4, Inc_d5, Inc_p5,
        Inc_a5, Inc_d6, Inc_p6, Inc_a6, Inc_d7, Inc_p7,
        Inc_a7, Inc_d8, Inc_p8, Inc_a8, Inc_d9, Inc_p9,
        Inc_a9, Inc_d10, Inc_p10, Inc_a10)
        key:  Call
```

DATA DICTIONARY

Data Name	Inc_d1
Aliases	None
Data type	Numeric
Format	5 numerics
Range	10000 = whole note
	07500 = dotted half note
	05000 = half note
	03750 = dotted quarter note
	02500 = quarter note
	01875 = dotted eighth note
	01250 = eighth note

00625 = sixteenth note

Responsibility
Security
Availability
Frequency
Dependencies
Comments Originally received from graphical interface

Data Name Inc_d2
Aliases None
Data type Numeric
Format 5 numerics
Range 10000 = whole note
07500 = dotted half note
05000 = half note
03750 = dotted quarter note
02500 = quarter note
01875 = dotted eighth note
01250 = eighth note
00625 = sixteenth note

Responsibility
Security
Availability
Frequency
Dependencies
Comments Originally received from graphical interface

Data Name Inc_d3
Aliases None
Data type Numeric
Format 5 numerics
Range 10000 = whole note
07500 = dotted half note
05000 = half note
03750 = dotted quarter note
02500 = quarter note
01875 = dotted eighth note
01250 = eighth note
00625 = sixteenth note

Responsibility
Security

Availability
Frequency
Dependencies
Comments — Originally received from graphical interface

Data Name — Inc_d4
Aliases — None
Data type — Numeric
Format — 5 numerics
Range — 10000 = whole note
07500 = dotted half note
05000 = half note
03750 = dotted quarter note
02500 = quarter note
01875 = dotted eighth note
01250 = eighth note
00625 = sixteenth note
Responsibility
Security
Availability
Frequency
Dependencies
Comments — Originally received from graphical interface

Data Name — Inc_d5
Aliases — None
Data type — Numeric
Format — 5 numerics
Range — 10000 = whole note
07500 = dotted half note
05000 = half note
03750 = dotted quarter note
02500 = quarter note
01875 = dotted eighth note
01250 = eighth note
00625 = sixteenth note
Responsibility
Security

Availability	
Frequency	
Dependencies	
Comments	Originally received from graphical interface

Data Name	Inc_d6
Aliases	None
Data type	Numeric
Format	5 numerics
Range	10000 = whole note 07500 = dotted half note 05000 = half note 03750 = dotted quarter note 02500 = quarter note 01875 = dotted eighth note 01250 = eighth note 00625 = sixteenth note
Responsibility	
Security	
Availability	
Frequency	
Dependencies	
Comments	Originally received from graphical interface

Data Name	Inc_d7
Aliases	None
Data type	Numeric
Format	5 numerics
Range	10000 = whole note 07500 = dotted half note 05000 = half note 03750 = dotted quarter note 02500 = quarter note 01875 = dotted eighth note 01250 = eighth note 00625 = sixteenth note
Responsibility	
Security	

Availability
Frequency
Dependencies
Comments — Originally received from graphical interface

Data Name — Inc_d8
Aliases — None
Data type — Numeric
Format — 5 numerics
Range
- 10000 = whole note
- 07500 = dotted half note
- 05000 = half note
- 03750 = dotted quarter note
- 02500 = quarter note
- 01875 = dotted eighth note
- 01250 = eighth note
- 00625 = sixteenth note

Responsibility
Security
Availability
Frequency
Dependencies
Comments — Originally received from graphical interface

Data Name — Inc_d9
Aliases — None
Data type — Numeric
Format — 5 numerics
Range
- 10000 = whole note
- 07500 = dotted half note
- 05000 = half note
- 03750 = dotted quarter note
- 02500 = quarter note
- 01875 = dotted eighth note
- 01250 = eighth note
- 00625 = sixteenth note

Responsibility

Security	
Availability	
Frequency	
Dependencies	
Comments	Originally received from graphical interface

Data Name	Inc_d10
Aliases	None
Data type	Numeric
Format	5 numerics
Range	10000 = whole note 07500 = dotted half note 05000 = half note 03750 = dotted quarter note 02500 = quarter note 01875 = dotted eighth note 01250 = eighth note 00625 = sixteenth note
Responsibility	
Security	
Availability	
Frequency	
Dependencies	
Comments	Originally received from graphical interface

Data Name	Inc_p1
Aliases	None
Data type	Numeric
Format	2 numerics
Range	00 = rest 90 = blank pitch 99 = bar 02 - 22 = notes, in even increments
Responsibility	
Security	

Availability	
Frequency	
Dependencies	
Comments	Originally received from graphical interface via temporary database

Data Name	Inc_p2
Aliases	None
Data type	Numeric
Format	2 numerics
Range	00 = rest 90 = blank pitch 99 = bar 02 - 22 = notes, in even increments
Responsibility	
Security	
Availability	
Frequency	
Dependencies	
Comments	Originally received from graphical interface via temporary database

Data Name	Inc_p3
Aliases	None
Data type	Numeric
Format	2 numerics
Range	00 = rest 90 = blank pitch 99 = bar 02 - 22 = notes, in even increments
Responsibility	
Security	
Availability	
Frequency	
Dependencies	
Comments	Originally received from graphical interface via temporary database

Data Name	Inc_p4
Aliases	None
Data type	Numeric
Format	2 numerics
Range	00 = rest 90 = blank pitch 99 = bar 02 - 22 = notes, in even increments
Responsibility	
Security	
Availability	
Frequency	
Dependencies	
Comments	Originally received from graphical interface via temporary database

Data Name	Inc_p5
Aliases	None
Data type	Numeric
Format	2 numerics
Range	00 = rest 90 = blank pitch 99 = bar 02 - 22 = notes, in even increments
Responsibility	
Security	
Availability	
Frequency	
Dependencies	
Comments	Originally received from graphical interface via temporary database

Data Name	Inc_p6
Aliases	None
Data type	Numeric
Format	2 numerics
Range	00 = rest 90 = blank pitch 99 = bar

	02 - 22 = notes, in even increments
Responsibility	
Security	
Availability	
Frequency	
Dependencies	
Comments	Originally received from graphical interface via temporary database

Data Name	Inc_p7
Aliases	None
Data type	Numeric
Format	2 numerics
Range	00 = rest 90 = blank pitch 99 = bar 02 - 22 = notes, in even increments
Responsibility	
Security	
Availability	
Frequency	
Dependencies	
Comments	Originally received from graphical interface via temporary database

Data Name	Inc_p8
Aliases	None
Data type	Numeric
Format	2 numerics
Range	00 = rest 90 = blank pitch 99 = bar 02 - 22 = notes, in even increments
Responsibility	
Security	
Availability	
Frequency	
Dependencies	
Comments	Originally received from

graphical interface via
temporary database

Data Name	Inc_p9
Aliases	None
Data type	Numeric
Format	2 numerics
Range	00 = rest 90 = blank pitch 99 = bar 02 - 22 = notes, in even increments
Responsibility	
Security	
Availability	
Frequency	
Dependencies	
Comments	Originally received from graphical interface via temporary database

Data Name	Inc_p10
Aliases	None
Data type	Numeric
Format	2 numerics
Range	00 = rest 90 = blank pitch 99 = bar 02 - 22 = notes, in even increments
Responsibility	
Security	
Availability	
Frequency	
Dependencies	
Comments	Originally received from graphical interface via temporary database

Data Name	Inc_a1
Aliases	None

Data type	Character
Format	1 character
Range	+ = sharp - = flat 0 = natural
Responsibility	
Security	
Availability	
Frequency	
Dependencies	
Comments	Originally received from graphical interface via temporary database

Data Name	Inc_a2
Aliases	None
Data type	Character
Format	1 character
Range	+ = sharp - = flat 0 = natural
Responsibility	
Security	
Availability	
Frequency	
Dependencies	
Comments	Originally received from graphical interface via temporary database

Data Name	Inc_a3
Aliases	None
Data type	Character
Format	1 character
Range	+ = sharp - = flat 0 = natural
Responsibility	
Security	
Availability	
Frequency	

Dependencies	
Comments	Originally received from graphical interface via temporary database

Data Name	Inc_a4
Aliases	None
Data type	Character
Format	1 character
Range	+ = sharp - = flat 0 = natural
Responsibility	
Security	
Availability	
Frequency	
Dependencies	
Comments	Originally received from graphical interface via temporary database

Data Name	Inc_a5
Aliases	None
Data type	Character
Format	1 character
Range	+ = sharp - = flat 0 = natural
Responsibility	
Security	
Availability	
Frequency	
Dependencies	
Comments	Originally received from graphical interface via temporary database

Data Name	Inc_a6
Aliases	None
Data type	Character
Format	1 character
Range	+ = sharp - = flat 0 = natural
Responsibility	
Security	
Availability	
Frequency	
Dependencies	
Comments	Originally received from graphical interface via temporary database

Data Name	Inc_a7
Aliases	None
Data type	Character
Format	1 character
Range	+ = sharp - = flat 0 = natural
Responsibility	
Security	
Availability	
Frequency	
Dependencies	
Comments	Originally received from graphical interface via temporary database

Data Name	Inc_a8
Aliases	None
Data type	Character
Format	1 character
Range	+ = sharp - = flat 0 = natural
Responsibility	
Security	
Availability	

Frequency	
Dependencies	
Comments	Originally received from graphical interface via temporary database

Data Name	Inc_a9
Aliases	None
Data type	Character
Format	1 character
Range	+ = sharp - = flat 0 = natural
Responsibility	
Security	
Availability	
Frequency	
Dependencies	
Comments	Originally received from graphical interface via temporary database

Data Name	Inc_a10
Aliases	None
Data type	Character
Format	1 character
Range	+ = sharp - = flat 0 = natural
Responsibility	
Security	
Availability	
Frequency	
Dependencies	
Comments	Originally received from graphical interface via temporary database

Data Name	Key
Aliases	
Data type	Character
Format	2 characters
Range	First character: A - G Second character: #, b, blank
Responsibility	
Security	
Availability	
Frequency	
Dependencies	
Comments	

Data Name	Call_num
Aliases	
Data type	Numeric
Format	5 numerics
Range	00000 through 99999
Responsibility	System-generated
Security	
Availability	
Frequency	
Dependencies	
Comments	Generated by the system for means of having unique number system

Data Name	Time1
Aliases	
Data type	Numeric
Format	1 digit
Range	1 - 9
Responsibility	Created by graphical interface
Security	
Availability	
Frequency	
Dependencies	
Comments	Denotes the first number in a time signature

Data Name	Time2
Aliases	
Data type	Numeric
Format	1 digit
Range	1 - 9
Responsibility	Created by graphical interface
Security	
Availability	
Frequency	
Dependencies	
Comments	Denotes the second number in a time signature

Data Name	MM
Aliases	
Data type	Character
Format	2characters
Range	blank = major m = minor
Responsibility	
Security	
Availability	
Frequency	
Dependencies	
Comments	Indicates major or minor key

```
*********************************************************
***
*                          AAA
*
*********************************************************
***

set procedure to main
do menu


*********************************************************
***
*                          ADDCOMP
*
*********************************************************
***

PROCEDURE ADDCOMP
**************************************
********** composer add subsystem**
**************************************
more = .t.
do while more
clear
*----------------Init memory variables
************ present composer add subsystem.
@ 2,1 say "RESEARCH MUSICOLOGICAL TOOL"
@ 2,60 say dtoc(date()) + "    " + time()
@ 3,0 to 3,79
@ 5,10 say "    DATABASE SYSTEM:  ADD COMPOSER
INFORMATION"
m_compf = space(10)
m_compi = space(1)
m_compl = space(20)
m_cloc = space(40)
m_bdate = space(4)
m_ddate = space(4)
@ 08,1 say "COMPOSER INFORMATION"
@ 10,01 say "Last name:" get m_compl picture "@!"
@ 11,01 say "First name:" get m_compf picture "@!"
@ 12,01 say "Initial:" get m_compi picture "@!"
read
error = .f.
if m_compl = space(20)
```

```
  error = .t.
endif (m_compl = space(20))
if .not. error
****before you do this must have opened the database,
****look for entered composer
select 3
use composer index c_lname
seek m_compl
if  found()
  if comp_init = m_compi .and. comp_fname = m_compf
    @ 22,1 say "This composer already exists on database"
    m_ans = .t.
    @ 23,1 say "Do you want to try again?(Y/N)" get m_ans
;
    picture "Y"
    read
    if .not. m_ans
      more = .f.
    endif (.not. m_ans)
  endif (comp_init = m_compi .and. comp_fname = m_compf)
endif (found())
if .not. found()
  @ 13,1 say "Birth year:"
  @ 13,15 get m_bdate
  @ 14,1 say "Death year:"
  @ 14,15 get m_ddate
  @ 15,1 say "Birthplace:"
  @ 15,15 get m_cloc picture "@!"
  read
  error = .f.
  if val(m_bdate) > 1989
     error = .t.
  endif (val(m_bdate) > 1989)
  if (val(m_ddate) < val(m_bdate) .or. val(m_ddate) >
1989) ;
     .and. m_ddate <> space(4)
     error = .t.
  endif (m_ddate checks)
  if .not. error
  append blank
  replace comp_fname with m_compf
  replace comp_init with m_compi
  replace comp_lname with m_compl
  replace comp_bdate with m_bdate
  replace comp_ddate with m_ddate
  replace comp_loc with m_cloc
```

```
  answer = .t.
  @ 23,1 say "Do you want to add another composer?" ;
      get answer picture "Y"
  read
  if .not. answer
    more = .f.
  endif (.not. answer)
 else
   @ 22,1 say "Error in date entry."
   m_ans = .t.
   @ 23,1 say "Do you want to try again?(Y/N)" get m_ans
;
     picture "Y"
   read
   if .not. m_ans
     more = .f.
   endif (.not. m_ans)
  endif (.not. error)
 else
  if comp_init <> m_compi .or. comp_fname <> m_compf
    @ 13,1 say "Birth year:"
    @ 13,15 get m_bdate
    @ 14,1 say "Death year:"
    @ 14,15 get m_ddate
    @ 15,1 say "Birthplace:"
    @ 15,15 get m_cloc picture "@!"
    read
    error = .f.
    if val(m_bdate) > 1989
      error = .t.
    endif (bdate checks)
    if (val(m_ddate) < val(m_bdate) .or. val(m_ddate) >
1989) ;
       .and. m_ddate <> space(4)
      error = .t.
    endif (ddate checks)
    if .not. error
    append blank
    replace comp_fname with m_compf
    replace comp_init with m_compi
    replace comp_lname with m_compl
    replace comp_bdate with m_bdate
    replace comp_ddate with m_ddate
    replace comp_loc with m_cloc
    answer = .t.
    @ 23,1 say "Do you want to add another composer?" ;
       get answer picture "Y"
    read
```

```
    if .not. answer
       more = .f.
    endif (.not. answer)
    else
     @ 22,1 say "Error in date entry."
     m_ans = .t.
     @ 23,1 say "Do you want to try again?(Y/N)" get
m_ans ;
       picture "Y"
     read
     if .not. m_ans
        more = .f.
     endif (.not. m_ans)
    endif (.not. error)

  endif (comp_init = m_compi .and. comp_fname = m_compf)
endif (.not. found())
else
  @ 22,1 say "Composer's last name must be greater than
spaces."
  m_ans = .t.
  @ 23,1 say "Do you want to try again(Y/N)?" get m_ans ;
    picture "Y"
  read
  if .not. m_ans
    more = .f.
  endif (.not. m_ans)
endif (.not. error)
enddo (do while more)
return
```

```
*************************************************************
***
*                              ADDINC
*
*************************************************************
***

PROCEDURE ADDINC
******** database add subsystem*****
more = .t.
do while more
clear
m_key = space(2)
*-----------------Create underline variable, uline.
************* present database add subsystem.
@ 2,1 say "RESEARCH MUSICOLOGICAL TOOL"
@ 2,60 say dtoc(date()) + "    " + time()
@ 3,0 to 3,79
@ 5,10 say "    DATABASE SYSTEM:  ADD MUSIC RECORD"
m_d1 = 0
m_p1 = 0
m_a1 = " "
m_d2 = 0
m_p2 = 0
m_a2 = " "
m_d3 = 0
m_p3 = 0
m_a3 = " "
m_d4 = 0
m_p4 = 0
m_a4 = " "
m_d5 = 0
m_p5 = 0
m_a5 = " "
m_d6 = 0
m_p6 = 0
m_a6 = " "
m_d7 = 0
m_p7 = 0
m_a7 = " "
m_d8 = 0
m_p8 = 0
m_a8 = " "
m_d9 = 0
m_p9 = 0
m_a9 = " "
m_d10 = 0
```

```
m_pl0 = 0
m_al0 = " "
temp_num = 00000
new_num = 00000
old_call = 00000
m_compl = space(20)
m_compf = space(10)
m_compi = space(1)
m_stitle = space(40)
c_link = .f.
s_link = .f.
select 1
use music index call_num

do while .not. EOF()
  skip
enddo (.not. EOF())
if .not. BOF()
  skip -1
  store call_num to temp_num
endif (.not. BOF())
temp_num = temp_num + 1
new_num = temp_num + 10000
@ 09,1 say "Please wait to input incipit for call #"
@ 09,41 say temp_num
erase  \tp\gfile1.doc
*erase \project\gfile1.doc

select 2
use graphf index callg
append blank
replace act with "A"
replace call with new_num
*------send file to graphics interface----------
copy to \tp\gfile1.doc sdf
*copy to \project\gfile1.doc sdf

*--------------delete record from graph database so won't
clutter
zap
*-----------------------------------------------
*---run executable version of graphics program
run cd \tp
run grapintf
run cd \demo
*--------retrieve file -------------------------
title = "    DATABASE SYSTEM:  ADD MUSIC RECORD"
do titles with title
```

```
select 2
use graphf index callg
append from \tp\gifile.dat sdf
*append from \project\gfile2.doc sdf

temp_call = call - 10000
store temp_call to m_call
store mm to m_mm
store time1 to m_time1
store time2 to m_time2
store key to m_key
store act to ret_act
do stnotes with m_d1, m_p1, m_a1, ;
m_d2, m_p2, m_a2, ;
m_d3, m_p3, m_a3, ;
m_d4, m_p4, m_a4, ;
m_d5, m_p5, m_a5, ;
m_d6, m_p6, m_a6, ;
m_d7, m_p7, m_a7, ;
m_d8, m_p8, m_a8, ;
m_d9, m_p9, m_a9, ;
m_d10, m_p10, m_a10
if ret_act = "A"
   okflag = .t.
   store 0 to difcter
   do infer with m_d1, m_p1, m_a1, ;
m_d2, m_p2, m_a2, ;
m_d3, m_p3, m_a3, ;
m_d4, m_p4, m_a4, ;
m_d5, m_p5, m_a5, ;
m_d6, m_p6, m_a6, ;
m_d7, m_p7, m_a7, ;
m_d8, m_p8, m_a8, ;
m_d9, m_p9, m_a9, ;
m_d10, m_p10, m_a10, ;
okflag, m_call, difcter, m_time1, m_time2
   if okflag
      m_title = space(80)
      @ 20, 1 say "Enter title of composition:" get
m_title ;
        picture "@!"
      read
      select 1
      use music index call_num
      append blank
      replace call_num with m_call
      store call_num to old_call
      replace key with m_key
```

```
      replace title with m_title
      replace time1 with m_time1
      replace time2 with m_time2
      do rpnotes with m_d1, m_p1, m_a1, ;
m_d2, m_p2, m_a2, ;
m_d3, m_p3, m_a3, ;
m_d4, m_p4, m_a4, ;
m_d5, m_p5, m_a5, ;
m_d6, m_p6, m_a6, ;
m_d7, m_p7, m_a7, ;
m_d8, m_p8, m_a8, ;
m_d9, m_p9, m_a9, ;
m_d10, m_p10, m_a10

  endif (okflag)
  close databases
   @ 22,1 clear
   @ 23,1 clear
  store .t. to m_ans
   @ 22,1 say "Do you wish to add a composer now?" get
m_ans ;
         picture "Y"
   read
   if m_ans
      if old_call = 00000
         @ 23,1 say "What call number do you wish to
use?" ;
           get old_call picture '99999'
           read
      endif (old_call = 00000)
      do inccadd with old_call, m_compl, m_compf,
m_compi, c_link
      close databases
   endif (m_ans)
   store .t. to m_ans
   @ 22,1 clear
   @ 22,1 say "Do you wish to add a source now?  " get
m_ans ;
    picture "Y"
   read
   if m_ans
     if old_call = 00000
        @ 23,1 say "What call number do you wish to use?"
;
        get old_call picture '99999'
        read
     endif (old_call = 00000)
      do incsadd with old_call, m_stitle, s_link
```

```
      close databases
   endif (m_ans)
   if c_link .and. s_link
      select 8
      use link index calll, cnamel, srcl
      append blank
      replace call_num with old_call
      replace comp_lname with m_compl
      replace comp_fname with m_compf
      replace comp_init with m_compi
      replace src_title with m_stitle
   endif (c_link .and. s_link)
else
   @ 20,1 say "Add is not indicated by graphics
interface"
 endif (ret_act = "A")
 answer = .t.
 @ 22,1 clear
 @ 22,1 say "Do you want to add another incipit?" ;
 get answer picture "Y"
 read
 if .not. answer
   more = .f.
 endif (.not. answer)
select 2
use graphf index callg
zap
enddo (more)
close databases
return
```

```
*********************************************************
***
*                              ADDSRC
*
*********************************************************
***

PROCEDURE ADDSRC
**************************************
*********** source add subsystem***
**************************************
more = .t.
do while more
clear
*-----------------Init memory variables
************* present rule type add subsystem.
@ 2,1 say "RESEARCH MUSICOLOGICAL TOOL"
@ 2,60 say dtoc(date()) + "    " + time()
@ 3,0 to 3,79
@ 5,10 say "    DATABASE SYSTEM:  ADD MUSIC SOURCE"
select 5
use source index sources
m_stitle = space(40)
m_stype = space(40)
m_sdate = space(4)
@ 08,1 say "SOURCE INFORMATION"
@ 10,1 say "Enter source:" get m_stitle picture "@!"
read
error = .f.
if m_stitle = space(40)
  error = .t.
endif (m_stitle = space())
if .not. error
****before you do this must have opened the database,
****look for entered source
seek m_stitle
if  found()
    @ 22,1 say "This source already exists"
    m_ans = .t.
    @ 23,1 say "Do you want to try again(Y/N)?" get m_ans
;
    picture "Y"
    read
    if .not. m_ans
      more = .f.
    endif (.not. m_ans)
endif (found())
```

```
if .not. found()
  @ 11,1 say "Source type:"
  @ 11,15 get m_stype  picture "@!"
  @ 12,1 say "Source date:"
  @ 12,15 get m_sdate
  read
  error = .f.
   if .not. (val(m_sdate) <= 1989)
     error = .t.
   endif (date checks)
  if .not. error
  append blank
  replace src_type with m_stype
  replace src_title with m_stitle
  replace src_date with m_sdate
  answer = .t.
  @ 23,1 say "Do you want to add another source?" ;
     get answer picture "Y"
  read
  if .not. answer
    more = .f.
  endif (.not. answer)
 else
  @ 22,1 say "Error in date entry."
  m_ans = .t.
  @ 23,1 say "Do you want to try again(Y/N)?" get m_ans ;
  picture "Y"
  read
  if m_ans
    more = .f.
  endif (.not. m_ans)
 endif (.not. error)
endif (.not. found())
else
  @ 22,1 say "Source title cannot be spaces."
  m_ans = .t.
  @ 23,1 say "Do you want to try again(Y/N)?" get m_ans ;
  picture "Y"
  read
  if .not. m_ans
     more = .f.
  endif (.not. m_ans)
endif (.not. error)
enddo (do while more)
return
```

```
************************************************************
***
*                          CHGNOTE2
*
************************************************************
***

PROCEDURE CHGNOTE2
parameters inc_p1, inc_a1, inc_p2, inc_a2, inc_p3,
inc_a3, ;
inc_p4, inc_a4, inc_p5, inc_a5, inc_p6, inc_a6, inc_p7, ;
inc_a7, inc_p8, inc_a8, inc_p9, inc_a9, inc_p10, ;
inc_a10, x_p1, x_p2, x_p3, x_p4, x_p5, x_p6, x_p7, x_p8,
x_p9, ;
x_p10
x_p1 = inc_p1
if inc_a1 = "+"
  x_p1 = x_p1 + 1
endif (+)
if inc_a1 = "-"
  x_p1 = x_p1 - 1
endif (-)

x_p2 = inc_p2
if inc_a2 = "+"
  x_p2 = x_p2 + 1
endif (+)
if inc_a2 = "-"
  x_p2 = x_p2 - 1
endif (-)

x_p3 = inc_p3
if inc_a3 = "+"
  x_p3 = x_p3 + 1
endif (+)
if inc_a3 = "-"
  x_p3 = x_p3 - 1
endif (-)

x_p4 = inc_p4
if inc_a4 = "+"
  x_p4 = x_p4 + 1
endif (+)
if inc_a4 = "-"
  x_p4 = x_p4 - 1
endif (-)
```

```
x_p5 = inc_p5
if inc_a5 = "+"
  x_p5 = x_p5 + 1
endif (+)
if inc_a5 = "-"
   x_p5 = x_p5 - 1
endif (-)

x_p6 = inc_p6
if inc_a6 = "+"
   x_p6 = x_p6 + 1
endif (+)
if inc_a6 = "-"
  x_p6 = x_p6 - 1
endif (-)

x_p7 = inc_p7
if inc_a7 = "+"
  x_p7 = x_p7 + 1
endif (+)
if inc_a7 = "-"
  x_p7 = x_p7 - 1
endif (-)

x_p8 = inc_p8
if inc_a8 = "+"
  x_p8 = x_p8 + 1
endif (+)
if inc_a8 = "-"
  x_p8 = x_p8 - 1
endif (-)

x_p9 = inc_p9
if inc_a9 = "+"
  x_p9 = x_p9 + 1
endif (+)
if inc_a9 = "-"
  x_p9 = x_p9 - 1
endif (-)

x_p10 = inc_p10
if inc_a10 = "+"
  x_p10 = x_p10 + 1
endif (+)
if inc_a10 = "-"
  x_p10 = x_p10 - 1
endif (-)
return
```

```
************************************************************
***
*                          CHGNOTES
*
************************************************************
***

PROCEDURE CHGNOTES
parameters m_p1, m_a1, m_p2, m_a2, m_p3, m_a3, m_p4,
m_a4, ;
m_p5, m_a5, m_p6, m_a6, m_p7, m_a7, m_p8, m_a8, m_p9,
m_a9, m_p10, ;
m_a10, c_p1, c_p2, c_p3, c_p4, c_p5, c_p6, c_p7, c_p8,
c_p9, ;
c_p10
c_p1 = m_p1
if m_a1 = "+"
   c_p1 = c_p1 + 1
endif (+)
if m_a1 = "-"
  c_p1 = c_p1 - 1
endif (-)

c_p2 = m_p2
if m_a2 = "+"
  c_p2 = c_p2 + 1
endif (+)
if m_a2 = "-"
  c_p2 = c_p2 - 1
endif (-)

c_p3 = m_p3
if m_a3 = "+"
  c_p3 = c_p3 + 1
endif (+)
if m_a3 = "-"
  c_p3 = c_p3 - 1
endif (-)

c_p4 = m_p4
if m_a4 = "+"
  c_p4 = c_p4 + 1
endif (+)
if m_a4 = "-"
  c_p4 = c_p4 - 1
endif (-)
```

```
c_p5 = m_p5
if m_a5 = "+"
  c_p5 = c_p5 + 1
endif (+)
if m_a5 = "-"
   c_p5 = c_p5 - 1
endif (-)

c_p6 = m_p6
if m_a6 = "+"
   c_p6 = c_p6 + 1
endif (+)
if m_a6 = "-"
  c_p6 = c_p6 - 1
endif (-)

c_p7 = m_p7
if m_a7 = "+"
  c_p7 = c_p7 + 1
endif (+)
if m_a7 = "-"
  c_p7 = c_p7 - 1
endif (-)

c_p8 = m_p8
if m_a8 = "+"
  c_p8 = c_p8 + 1
endif (+)
if m_a8 = "-"
  c_p8 = c_p8 - 1
endif (-)

c_p9 = m_p9
if m_a9 = "+"
  c_p9 = c_p9 + 1
endif (+)
if m_a9 = "-"
  c_p9 = c_p9 - 1
endif (-)

c_p10 = m_p10
if m_a10 = "+"
  c_p10 = c_p10 + 1
endif (+)
if m_a10 = "-"
  c_p10 = c_p10 - 1
endif (-)
```

```
return

*********************************************************
***
*                          DBADD
*
*********************************************************
***

PROCEDURE DBADD
**************************************
********** dbadd menu*********
**************************************
************ present add menu and get user's choice.
choice = 0
do while choice # 4
    clear
    @ 2,1 say "RESEARCH MUSICOLOGICAL TOOL"
    @ 2,60 say dtoc(date()) + "    " + time()
    @ 3,0 to 3,79
    ?
    ?
    text
                DATABASE ADD MENU

                Available selections include:


                1.  Add An Incipit/Schedule Expert
                2.  Add A Source Record
                3.  Add A Composer Record
                4.  Return To Previous Screen
    endtext
    @ 20,1 say "Enter choice " get choice;
      picture "9" range 1,4
    read
*---------------------------------branch to appropriate
program.
    if choice < 1 .or. choice > 4
       @ 23,1 say "CHOICE MUST BE WITHIN RANGE OF 1-4 "
       @ 24,1 say "PRESS ANY KEY TO CONTINUE"
       read
    else
       do case
            case choice = 1
                do addinc
            case choice = 2
```

```
            do addsrc
        case choice = 3
            do addcomp
    endcase
endif (choice < 1 .or. choice > 4)

enddo (while choice # 4)
close databases
return


*********************************************************
***
*                          DBDEL
*
*********************************************************
***

PROCEDURE DBDEL
**************************************
********** dbdelete menu*********
**************************************
*-----------------Create underline variable, uline.
************ present delete menu and get user's choice.
choice = 0
do while choice # 4
    clear
    @ 2,1 say "RESEARCH MUSICOLOGICAL TOOL"
    @ 2,60 say dtoc(date()) + "    " + time()
    @ 3,0 to 3,79
    ?
    ?
    text
                DATABASE DELETE MENU

                Available selections include:


                1.  Delete An Incipit/Schedule Expert
                2.  Delete A Source Record
                3.  Delete A Composer Record
                4.  Return To Previous Screen
    endtext
    @ 20,1 say "Enter choice " get choice;
      picture "9" range 1,4
    read
*---------------------------------branch to appropriate
program.
```

```
    if choice < 1 .or. choice > 4
       @ 23,1 say "CHOICE MUST BE WITHIN RANGE OF 1-4 "
       @ 24,1 say "PRESS ANY KEY TO CONTINUE"
       read
    else
       do case
           case choice = 1
               do delinc
           case choice = 2
               do delsrc
           case choice = 3
               do delcomp
       endcase
    endif (choice < 1 .or. choice > 4)

enddo (while choice # 4)
close databases
return


************************************************************
***
*                          DBFAC
*
************************************************************
***

PROCEDURE DBFAC
    ***************************************
    ********** database facility menu*****
    ***************************************
    *----------------Create underline variable, uline.
    ************* present database submenu   and get
user's choice.
    choice = 0
    do while choice # 5
        clear
       @ 2,1 say "RESEARCH MUSICOLOGICAL TOOL"
       @ 2,60 say dtoc(date()) + "    " + time()
       @ 3,0 to 3,79
        ?
        text
                DATABASE FACILITY

                Available database functions include:

                1.  Add a new record
                2.  Update an existing record
```

```
                3.  Delete an existing record
                4.  Query
                5.  Exit to previous screen
        endtext
        @ 20,1 say "Enter choice (1-5) " get choice;
         picture "9" range 1,5
        read
        *----------------------------branch to
appropriate program.
        if choice < 1 .or. choice > 5
           @ 23,1 say "CHOICE MUST BE WITHIN RANGE OF 1-
5"
           @ 24,1 say "PRESS ANY KEY TO CONTINUE"
           read
        else
           do case
           case choice = 1
               do dbadd
           case choice = 2
               do dbupd
           case choice = 3
               do dbdel
           case choice = 4
               do dbquery
          endcase
         endif (choice < 1 .or. choice > 5)

    enddo (while choice # 5)
    close databases
    return
```

```
*********************************************************
***
*                          DBQUERY
*
*********************************************************
***

PROCEDURE DBQUERY
**************************************
********** database query subsystem***
**************************************
temp_num1 = 0
@ 22,1 say "Enter call number for inquiry:" get temp_num1
temp_num = temp_num1
*--------set up loop for query
more = .t.
first = .t.
do while more
clear
************ present database add subsystem.
@ 2,1 say "RESEARCH MUSICOLOGICAL TOOL"
@ 2,60 say dtoc(date()) + "    " + time()
@ 3,0 to 3,79
@ 5,10 say "    DATABASE SYSTEM:  QUERY EXISTING RECORD"
*------------initialize temp_num to 0
select 1
use music index call_num
seek temp_num
if .not. found()
  if .not. first
   @ 22,1 say "This call number is not found"
   m_ans = .t.
   @ 23,1 say "Do you want to try again(Y/N)?" get m_ans
;
   picture "Y"
   read
   if .not. m_ans
     more = .f.
   endif (.not. m_ans)
  endif (.not. first)
endif (.not. found())
if found()
@ 08,1 say "COMPOSITION INFORMATION"
@ 09,1 say "Call number:"
@ 09,15 say call_num
@ 09,30 say "Key signature:"
@ 09,45 say key
```

```
@ 10,1 say "Title:"
@ 10,15 say title
answer = .t.
store call_num to m_call_num
@ 22,1 say "Do you wish to see sources (Y/N)?" get answer
;
  picture "Y"
read
if answer
   select 6
   use refer_in index callr, srcr
   seek m_call_num
   if found()
     @ 12,1 say "SOURCE INFORMATION"
     list while call_num = m_call_num
   else
     @ 24,1 say "No sources have been added for this
composition"
   endif (found())
 endif (answer)
answer = .t.
@ 22,1 say "Do you wish to see composers (Y/N)?" get
answer ;
  picture "Y"
read
if answer
   select 4
   use written index callw, cnamew
   seek m_call_num
   if found()
      @ 12,1 say "COMPOSER INFORMATION"
      list while call_num = m_call_num
   else
      @ 24,1 say "No composers have added for this
composition"
  endif (found())
 endif (answer)
endif (found())
m_ans = .t.
if .not. first
  @ 22,1  say "Do you wish to inquire again?" get m_ans
picture "Y"
  read
else
  first = .f.
endif (.not. first)
if m_ans
  @ 22,1 clear
```

```
  @ 22,1 say "Enter next call number for inquiry:" get
temp_num
  read
else
  more = .f.
endif (m_ans)
enddo (more)
return


*********************************************************
***
*                          DBUPD
*
*********************************************************
***

PROCEDURE DBUPD
**************************************
********** dbupdate menu*********
**************************************
*-----------------Create underline variable, uline.
************ present update menu and get user's choice.
choice = 0
do while choice # 4
    clear
    @ 2,1 say "RESEARCH MUSICOLOGICAL TOOL"
    @ 2,60 say dtoc(date()) + "    " + time()
    @ 3,0 to 3,79
    ?
    ?
    text
                DATABASE UPDATE MENU

                Available selections include:


                1.  Update An Incipit/Schedule Expert
                2.  Update A Source Record
                3.  Update A Composer Record
                4.  Return To Previous Screen
    endtext
    @ 20,1 say "Enter choice " get choice;
      picture "9" range 1,4
    read
*---------------------------------branch to appropriate
program.
    if choice < 1 .or. choice > 4
```

```
      @ 23,1 say "CHOICE MUST BE WITHIN RANGE OF 1-4 "
      @ 24,1 say "PRESS ANY KEY TO CONTINUE"
      read
   else
      do case
          case choice = 1
              do updinc
          case choice = 2
              do updsrc
          case choice = 3
              do updcomp
      endcase
   endif (choice < 1 .or. choice > 4)

enddo (while choice # 4)
close databases
return


*********************************************************
***
*                          DELCOMP
*
*********************************************************
***

PROCEDURE DELCOMP
**************************************
*********** composer delete subsystem**
**************************************
more = .t.
do while more
clear
*-----------------Init memory variables
************ present composer delete subsystem.
@ 2,1 say "RESEARCH MUSICOLOGICAL TOOL"
@ 2,60 say dtoc(date()) + "    " + time()
@ 3,0 to 3,79
@ 5,10 say "    DATABASE SYSTEM:  DELETE COMPOSER
INFORMATION"
m_compf = space(10)
m_compi = space(1)
m_compl = space(20)
@ 08,1 say "COMPOSER INFORMATION"
@ 10,1 say "Last name:" get m_compl picture "@!"
@ 11,1 say "First name:" get m_compf picture "@!"
@ 12,1 say "Initial:" get m_compi picture "@!"
read
```

```
select 3
use composer index c_lname
seek m_compl
if .not. found()
    @ 22,1 say "This composer does not exist on database"
    m_ans = .t.
    @ 23,1 say "Do you want to try again(Y/N)?" get m_ans
;
    picture "Y"
    read
    if .not. m_ans
       more = .f.
    endif (.not. m_ans)
endif (.not. found())
if found()
   if comp_init = m_compi .and. comp_fname = m_compf
     store comp_bdate to m_bdate
     store comp_ddate to m_ddate
     store comp_loc to m_cloc
     @ 13,1 say "Birth date:"
     @ 13,15 say m_bdate
     @ 14,1 say "Death date:"
     @ 14,15 say m_ddate
     @ 15,1 say "Birthplace:"
     @ 15,15 say m_cloc
     store .f. to m_ans
     @ 19,1 say "Delete this composer?" get m_ans picture
"Y"
     read
     if m_ans
       recnumb = recno()
       delete record recnumb
       pack
*------now delete all written records with this composer-
-----
       select 4
       use written index cnamew, callw
       delete for comp_lname = m_compl .and. ;
                  comp_fname = m_compf .and. ;
                  comp_init = m_compi
       pack
       select 8
       use link index cnamel, calll, srcl
       delete for comp_lname = m_compl .and. ;
                  comp_fname = m_compf .and. ;
                  comp_init = m_compi
       pack
     endif (m_ans)
```

```
     answer = .t.
     @ 23,1 say "Do you want to delete another composer?"
;
       get answer picture "Y"
     read
     if .not. answer
       more = .f.
     endif (.not. answer)
   else
     @ 22,1 say "This composer does not exist on
database."
     m_ans = .t.
     @ 23,1 say "Do you want to try again(Y/N)?" get
m_ans ;
     picture "Y"
     read
     if .not. m_ans
        more = .f.
     endif (.not. m_ans)
   endif (comp_init = m_compi .and. comp_fname = m_compf)
endif ( found())
enddo (do while more)
close databases
return
```

```
*********************************************************
***
*                          DELINC
*
*********************************************************
***

PROCEDURE DELINC
********database incipit delete****
more = .t.
do while more
clear
*-----------------Init memory variables
*---in order to delete a music record---
*------1  schedule graphics interface with  call #
*------   and activity indicator  - "D" for delete
*------2  graphics interface displays incipit
*------3  if want to delete this one
*------------delete the record
*----------else
*------------try another call number???
*-------------------------------------------------------
--------
m_call = 00000
new_num = 00000
************* present incipit delete subsystem.
@ 2,1 say "RESEARCH MUSICOLOGICAL TOOL"
@ 2,60 say dtoc(date()) + "    " + time()
@ 3,0 to 3,79
@ 5,10 say "    DATABASE SYSTEM:  DELETE MUSIC RECORD"
m_d1 = 0
m_p1 = 0
m_a1 = " "
m_d2 = 0
m_p2 = 0
m_a2 = " "
m_d3 = 0
m_p3 = 0
m_a3 = " "
m_d4 = 0
m_p4 = 0
m_a4 = " "
m_d5 = 0
m_p5 = 0
m_a5 = " "
m_d6 = 0
```

```
m_p6 = 0
m_a6 = " "
m_d7 = 0
m_p7 = 0
m_a7 = " "
m_d8 = 0
m_p8 = 0
m_a8 = " "
m_d9 = 0
m_p9 = 0
m_a9 = " "
m_d10 = 0
m_p10 = 0
m_a10 = " "
select 1
use music index call_num
@ 10,1 say "Enter the call number you wish to delete:" ;
  get m_call picture '99999'
read
seek m_call
if .not. found()
   @ 22,1 say "This call number does not exist"
   m_ans = .t.
   @ 23,1 say "Do you want to try again(Y/N)?" get m_ans
;
   picture "Y"
   read
   if .not. m_ans
     more = .f.
   endif (.not. m_ans)
endif (.not. found())
if found()
  store title to m_title
  store time1 to m_time1
  store time2 to m_time2
  store key to m_key
  store mm to m_mm
  do stnotes with m_d1, m_p1, m_a1, ;
m_d2, m_p2, m_a2, ;
m_d3, m_p3, m_a3, ;
m_d4, m_p4, m_a4, ;
m_d5, m_p5, m_a5, ;
m_d6, m_p6, m_a6, ;
m_d7, m_p7, m_a7, ;
m_d8, m_p8, m_a8, ;
m_d9, m_p9, m_a9, ;
m_d10, m_p10, m_a10
   @ 22,1 say "Please wait to verify incipit"
```

```
   select 2
   use graphf index callg
   append blank
   replace act with "D"
   new_num = m_call + 10000
   replace call with new_num
   replace key with m_key
   replace mm with m_mm
   replace time1 with m_time1
   replace time2 with m_time2
   do rpnotes with m_d1, m_p1, m_a1, ;
m_d2, m_p2, m_a2, ;
m_d3, m_p3, m_a3, ;
m_d4, m_p4, m_a4, ;
m_d5, m_p5, m_a5, ;
m_d6, m_p6, m_a6, ;
m_d7, m_p7, m_a7, ;
m_d8, m_p8, m_a8, ;
m_d9, m_p9, m_a9, ;
m_d10, m_p10, m_a10
*----send file to graphic interface-----
*  erase \project\gfile1.doc
   erase \tp\gfile1.doc
*  copy to gfile1.doc sdf
   copy to \tp\gfile1.doc sdf
*-----SIMULATE GRAPHICS INTERFACE HERE
run cd \tp
run grapintf
run cd \demo
title = "    DATABASE SYSTEM:  DELETE MUSIC RECORD"
do titles with title
close databases
store .t. to m_ans
@ 22,1 clear
@ 22,1 say "Is this the incipit you wanted to delete?" ;
get m_ans picture "Y"
read
if m_ans
  select 1
  use music index call_num
  seek m_call
  delete for call_num = m_call
  pack
  close databases
*----now delete any written or refer_in segments for call
#----
  select 4
  use written index callw, cnamew
```

```
  seek m_call
  delete for call_num = m_call
  pack
  close databases
  select 6
  use refer_in index callr, srcr
  seek m_call
  delete for call_num = m_call
  pack
  close databases
  select 8
  use link index calll, cnamel, srcl
  seek m_call
  delete for call_num = m_call
  pack
  close databases
endif (m_ans)
answer = .t.
 @ 22,1 clear
 @ 22,1 say "Do you want to delete another incipit?" ;
 get answer picture "Y"
 read
 if .not. answer
   more = .f.
 endif (.not. answer)
endif (found())
select 2
use graphf index callg
zap
enddo (more)
close databases
return
```

```
*************************************************************
***
*                              DELSRC
*
*************************************************************
***

PROCEDURE DELSRC
**************************************
*********** source delete subsystem***
**************************************
more = .t.
do while more
clear
*-----------------Init memory variables
************* present source delete subsystem.
@ 2,1 say "RESEARCH MUSICOLOGICAL TOOL"
@ 2,60 say dtoc(date()) + "    " + time()
@ 3,0 to 3,79
@ 5,10 say "    DATABASE SYSTEM:  DELETE MUSIC SOURCE"
m_stitle = space(40)
select 1
use source index sources
@ 08,1 say "SOURCE INFORMATION"
@ 10,1 say "Enter source:" get m_stitle picture "@!"
read
****before you do this must have opened the database,
****look for entered source
seek m_stitle
if .not. found()
    @ 22,1 say "This source does not exist"
    m_ans = .t.
    @ 23,1 say "Do you want to try again(Y/N)?" get m_ans
;
    picture "Y"
    read
    if .not. m_ans
       more = .f.
    endif (.not. m_ans)
endif (.not. found())
if found()
    store src_type to m_stype
    store src_date to m_sdate
    store src_title to m_stitle
  @ 11,1 say "Source type:"
  @ 11,15 say m_stype
  @ 12,1 say "Source date:"
```

```
  @ 12,15 say m_sdate
  store .f. to m_ans
  @ 22,1 say "Delete this source?" get m_ans picture "Y"
  read
  if m_ans
    delete for src_title = m_stitle
    pack
*---now delete all refer_in records with this source-----
    select 6
    use refer_in index srcr, callr
    delete all for src_title = m_stitle
    pack
    select 8
    use link index srcl, calll, cnamel
    delete all for src_title = m_stitle
    pack
  endif (m_ans )
  answer = .t.
  @ 22,1 say "Do you want to delete another source?" get
answer ;
    picture "Y"
  read
  if .not. answer
    more = .f.
  endif (.not. answer )
endif (.not. found())
enddo (do while more)
close databases
return
```

```
*************************************************************
***
*                          ESACT
*
*************************************************************
***

PROCEDURE ESACT
****************************************
********** rule type activate subsystem***
****************************************
more = .t.
do while more
clear
*-----------------Init memory variables
************ present rule type active/inactive*****
@ 2,1 say "RESEARCH MUSICOLOGICAL TOOL"
@ 2,60 say dtoc(date()) + "    " + time()
@ 3,0 to 3, 79
@ 5,10 say "    EXPERT SYSTEM UTILITY:  ACTIVATE RULE
TYPE"
m_type = space(20)
m_act_flag1 = .t.
select 7
use rule index types, num1, num2
@ 8,1 say "The following rules exist as active/inactive:"
@ 9,1 say "Rule"
@ 9,20 say "Act/Inact"
@ 9,30 say "Rule"
@ 9,50 say "Act/Inact"
prcter = 11
testtype = space(20)
do while prcter < 18 .and. .not. EOF()
  if type <> testtype
     @ prcter,1 say type
     @ prcter, 20 say act_flag1 picture "Y"
     store type to testtype
     do while type = testtype .and. .not. EOF()
       skip
     enddo (type = testtype .and. .not. EOF())
     if .not. EOF()
        @ prcter,30 say type
        @ prcter,50 say act_flag1 picture "Y"
        store type to testtype
     endif (.not. EOF())
     prcter = prcter + 1
   endif (type <> testtype)
```

```
   if .not. EOF()
     skip
   endif (.not. EOF())
enddo (prcter < 18 .and. .not. EOF())
@ 20,1 say "Enter rule type:"
@ 20,20 get m_type picture "@!"
read
****before you do this must open the database,
****look for entered rule
select 7
reindex
seek m_type
 if .not. found()
    m_ans = .t.
    @ 22,1 say "This rule does not exist"
    @ 23,1 say "Do you wish to try again(Y/N)?" get m_ans
;
    picture "Y"
    read
    if .not. m_ans
      more = .f.
    endif (m_ans)
 else
  store act_flag1 to m_act_flag1
  @ 21,1 say "Activate this rule type (Y/N)?:"
  @ 21,35 get m_act_flag1 picture "Y"
  read
  store .f. to m_ans
  @ 22,1 say "Make these changes?" get m_ans picture "Y"
  read
  if m_ans
      replace  act_flag1 with  m_act_flag1 for type =
m_type
  endif (m_ans)
  answer = .t.
  @ 23,1 say "Do you want to activate another rule type?"
;
    get answer picture "Y"
  read
  if .not. answer
   more = .f.
  endif (answer)
endif (.not.found())
enddo (do while more)
close databases
return
```

```
*************************************************************
***
*                          ESADD
*
*************************************************************
***

PROCEDURE ESADD
**************************************
*********** rule type add subsystem***
**************************************
more = .t.
do while more
clear
*-----------------Init memory variables
************* present rule type add subsystem.
@ 2,1 say "RESEARCH MUSICOLOGICAL TOOL"
@ 2,60 say dtoc(date()) + "    " + time()
@ 3,0 to 3,79
@ 5,10 say "  EXPERT SYSTEM UTILITY:  ADD RULE TYPE"
m_type = space(20)
m_act_flag1 = .t.
select 7
use rule index types, num1, num2
@ 8,1 say "The following rule types already exist:"
prcter = 9

testtype = space(20)
do while prcter < 18 .and. .not. EOF()
  if type <> testtype
    @ prcter,1 say type
    store type to testtype
    do while type = testtype .and. .not. EOF()
      skip
    enddo (type = testtype .and. .not. EOF()
    if .not. EOF()
      @ prcter,40 say type
      store type to testtype
    endif (.not. EOF()
    prcter = prcter + 1
   endif (type <> testtype)
  if .not. EOF()
  skip
  endif (.not. EOF())
enddo (prcter < 18 .and. .not. EOF())
m_ans = .t.
@ 18,1 say "Do you want to add another rule type(Y/N)?" ;
```

```
   get m_ans picture "Y"
   read
if m_ans
  @ 19,1 say "Enter rule type to add:"  get m_type
picture "@!"
  read
  error = .f.
  if m_type = space(20)
    error = .t.
  endif (m_type = space(20)
  if .not. error
****before you do this must open the database,
****look for entered rule
  reindex
  seek m_type
  if  found()
    @ 22,1 say "This rule already exists"
    m_ans = .t.
    @ 23,1 say  "Do you want to try again(Y/N)?"  get
m_ans ;
      picture "Y"
    read
    if .not. m_ans
      more = .f.
    endif (.not. m_ans)
  else
  @ 20,1 say "Activate this rule type (Y/N)?:" ;
       get m_act_flag1 picture "Y"
  read
  store .t. to m_act_flag2
  set order to 2
  go bottom
  if EOF()
    store 0 to last_seq
  else
    store seq_num1 to last_seq
  endif (EOF())
  next_seq = last_seq + 1
  store next_seq to m_seq_num1
  store 000 to m_seq_num2
  @ 21,1 say "Sequence of rule will default to last."
  append blank
  replace act_flag1 with m_act_flag1
  replace act_flag2 with m_act_flag2
  replace seq_num1 with m_seq_num1
  replace seq_num2 with m_seq_num2
  replace type with m_type
  answer = .t.
```

```
  @ 22,1 say "Do you want to add another rule type?" ;
    get answer picture "Y"
  read
  if .not. answer
    more = .f.
  endif (answer )
 endif (found())
  else
  @ 22,1 say "Rule type must be greater than spaces."
    m_ans = .t.
    @ 23,1 say "Do you want to try again(Y/N)?" get m_ans
;
     picture "Y"
    read
    if .not. m_ans
       more = .f.
    endif (.not. m_ans)
  endif (.not. error)
else
  more = .f.
endif (m_ans)
enddo (do while more)
close databases
return
```

```
*********************************************************
***
*                          ESMAIN
*
*********************************************************
***

PROCEDURE ESMAIN
*****************************************************
********** expert system maintenance subsystem******
*****************************************************
*-----------------Create underline variable, uline.
************* present expert system maintenance submenu.
choice = 0
do while choice # 6
    clear
    @ 2,1 say "RESEARCH MUSICOLOGICAL TOOL"
    @ 2,60 say dtoc(date()) + "    " + time()
    @ 3,0 to 3,79
    ?
    text
                EXPERT SYSTEM MAINTENANCE

                Available expert system functions
include:

                1.  Add a new rule type
                2.  Add a new rule within an existing
type  ..
                3.  Activate/Inactivate existing rule
type
                4.  Activate/Inactivate existing rules
within a type
                5.  Reorganize existing rules
                6.  Exit to previous screen
    endtext
    @ 20,1 say "Enter choice (1-6) " get choice;
      picture "9" range 1,6
    read
*---------------------------------branch to appropriate
program.
    if choice < 1 .or. choice > 6
       @ 23,1 say "CHOICE MUST BE WITHIN RANGE OF 1-6"
       @ 24,1 say "PRESS ANY KEY TO CONTINUE"
       read
  . else
       do case
```

```
            case choice = 1
                do esadd
            case choice = 2
                do esradd
            case choice = 3
                do esact
            case choice = 4
                do esract
            case choice = 5
                do esreorg
        endcase
    endif (choice < 1 .or. choice > 6)

enddo (while choice # 6)
close databases
return


*********************************************************
***
*                          ESRACT
*
*********************************************************
***

PROCEDURE ESRACT
**************************************
********** rule activate subsystem***
**************************************
more = .t.
do while more
clear
*----------------Init memory variables
************ present rule activate subsystem.
@ 2,1 say "RESEARCH MUSICOLOGICAL TOOL"
@ 2,60 say dtoc(date()) + "    " + time()
@ 3,0  to 3,79
@ 5,10 say  " EXPERT SYSTEM UTILITY:  ACTIVATE RULES
WITHIN TYPE"
m_type = space(20)
m_act_flag2 = .t.
@ 08,1 say "Enter rule type:"
@ 08,20 get m_type picture "@!"
read
****before you do this must open the database,
****look for entered rule
select 7
use rule index types, num1, num2
```

```
reindex
seek m_type
if .not. found()
    m_ans = .t.
    @ 22,1 say "This rule does not exist"
    @ 23,1 say "Do you want to try again(Y/N)?" get m_ans
;
    picture "Y"
    read
    if .not. m_ans
      more = .f.
    endif (.not. m_ans)
else
do while .not. EOF() .and. type = m_type
 if seq_num2 = 0
  skip
 else
  store act_flag2 to m_act_flag
  store field1 to m_field1
  store oper to m_oper
  store field2 to m_field2
  @ 12,1 say  "Rule field 1:"
  @ 12,17 say m_field1
  @ 13,1 say "Rule operator:"
  @ 13,17 say m_oper
  @ 14,1 say "Rule field 2:"
  @ 14,17 say m_field2
  @ 16,1 say "Activate this rule (Y/N)?:"  get
m_act_flag2   picture "Y"
  read
  store .f. to m_ans
  @ 22,1 say "Make these changes?" get m_ans picture "Y"
  read
  if m_ans
    replace act_flag2 with m_act_flag2
  endif (m_ans)
  skip
 endif (seq_num = 0)
enddo (type = m_type .and. .not. EOF())
answer = .t.
@ 22,1 say "Do you want to activate rules within another;
 type?" get answer picture "Y"
read
if .not. answer
   more = .f.
endif (answer)
endif (found())
enddo (do while more)
```

```
close databases
return


*********************************************************
***
*                          ESRADD
*
*********************************************************
***

PROCEDURE ESRADD
**************************************
********** rule add subsystem***
**************************************
more = .t.
do while more
clear
*-----------------Init memory variables
************ present rule add subsystem.
@ 2,1 say "RESEARCH MUSICOLOGICAL TOOL"
@ 2,60 say dtoc(date()) + "    " + time()
@ 3,0  to 3,79
@ 5,10 say "    EXPERT SYSTEM UTILITY:  ADD RULE WITHIN
TYPE"
m_type = space(20)
m_act_flag2 = .t.
select 7
use rule index types, num1, num2
@ 8,1 say "The following rule types exist:"
prcter = 9
testtype = space(20)
do while prcter < 18 .and. .not. EOF()
  if type <> testtype
     @ prcter,1 say type
     store type to testtype
     do while type = testtype .and. .not. EOF()
       skip
     enddo (type = testtype .and. .not. EOF())
     if .not. EOF()
        @ prcter, 40 say type
        store type to testtype
     endif (.not. EOF())
  prcter = prcter + 1
  endif (type <> testtype)
  if .not. EOF()
     skip
  endif (.not. EOF())
```

```
enddo (prcter < 18 .and. .not. EOF())
m_ans = .t.
@ 18,1 say "Enter rule type:"
@ 18,20 get m_type picture "@!"
read
****before you do this must open the database,
****look for entered rule
reindex
seek m_type
if .not. found()
    @ 22,1 say "This rule does not exist"
    m_ans = .t.
    @ 23,1 say "Do you want to try again(Y/N)?" get m_ans
;
    picture "Y"
    read
    if .not. m_ans
      more = .f.
    endif (.not. m_ans)
else
  clrcter = 7
  do while clrcter < 22
     @ clrcter,0 clear
     clrcter = clrcter + 1
  enddo (clrcter < 22)
  store act_flag1 to m_act_flag1
  store seq_num1 to m_seq_num1
  m_field1 = space(39)
  m_oper = space(2)
  m_field2 = space(39)
  @ 11,1 to 11,50
  @ 12,1 say "Valid entries for rule fields are:"
  @ 13,1 say "'rhydif'   'notedif'    'timedif'   any
number"
  @ 14,1 say "Valid entries for rule operator are:"
  @ 15,1 say "<>     =     <=     >=  "
  @ 16,1 say "A sample entry would be "
  @ 17,1 say "Field 1 :   notedif"
  @ 18,1 say "Operator:   >="
  @ 19,1 say "Field 2 :   5"
  @ 20,1 say "which prevents adding an incipit matching
by ;
more than 5 notes."
  error = .f.
  @ 07,1 say "Rule type = "
  @ 07,14 say m_type
  @ 08,1 say "Enter desired rule field 1:" ;
     get m_field1 picture "@!"
```

```
  @ 09,1 say "Enter desired rule operator:" ;
      get m_oper picture "@!"
  @ 10,1 say "Enter desired rule field 2:" ;
      get m_field2 picture "@!"
  read
****must be sure rule hasn't already been entered***
  error = .f.
do while type = m_type .and. .not. EOF()
   if field1 = m_field1 .and. oper = m_oper ;
.and. field2 = m_field2 .and. seq_num2 > 0
   exist = seq_num2
   error = .t.
   endif (checks for duplicates)
   skip
enddo (type = m_type .and. .not. EOF())
if .not. error

  if .not. (m_field1 = "RHYDIF" .or. m_field1 = "NOTEDIF"
;
 .or. m_field1 = "TIMEDIF" .or. (m_field1 >= "0" ;
 .and. m_field1 <= "9999999999"))
    error = .t.
  endif (m_field1 checks)
  if .not. (m_field2 = "RHYDIF" .or. m_field2 = "NOTEDIF"
;
 .or. m_field2 = "TIMEDIF" .or. (m_field2 >= "0" ;
 .and. m_field2 <= "9999999999"))
   error = .t.
  endif (m_field2 checks)
  if .not. (m_oper = "=" .or. m_oper = "<=" .or. ;
  m_oper = ">=" .or. m_oper = "<>" .or. m_oper = "<" .or.
;
  m_oper = ">")
   error = .t.
  endif (m_oper checks)
  if .not. error
   clrcter = 11
   do while clrcter < 24
     @ clrcter,0 clear
     clrcter = clrcter + 1
   enddo (clrcter < 24)
  @ 11,1 say "Activate this rule type (Y/N)?:"  get
m_act_flag2 ;
    picture "Y"
  read
  do while type = m_type .and. .not. EOF()
    skip
  enddo (type = m_type .and. .not. EOF())
```

```
  skip -1
  next_seq = seq_num2 + 1
  store next_seq to m_seq_num2
  @ 12,01 say "Sequence of rule will default to last."
  append blank
  replace act_flag1 with m_act_flag1
  replace act_flag2 with m_act_flag2
  replace seq_num1 with m_seq_num1
  replace seq_num2 with m_seq_num2
  replace type with m_type
  replace field1 with m_field1
  replace oper with m_oper
  replace field2 with m_field2
  answer = .t.
  @ 22,1 say "Do you want to add another rule?:" get
answer ;
    picture "Y"
  read
  if .not. answer
   more = .f.
  endif (answer)
else
  @ 22,1 clear
  @ 23,1 clear
  @ 22,1 say "Error in specifying fields."
  m_ans = .t.
  @ 23,1 say "Do you want to try again(Y/N)?" get m_ans ;
   picture "Y"
  read
  if .not. m_ans
    more = .f.
  endif (.not. m_ans)
 endif (.not. error)
else
  @ 21,1 clear
  ? " This rule already exists as rule #", exist
  m_ans = .t.
  @ 23,1 say "Do you want to try again(Y/N)?" get m_ans ;
  picture "Y"
  read
  if .not. m_ans
     more = .f.
  endif (.not. m_ans)
endif (.not. error)
endif (found())
enddo (more)
close databases
return
```

```
*********************************************************
***
*                          ESREORG
*
*********************************************************
***

PROCEDURE ESREORG
*************************************************
********** expert system reorganization subsystem***
*************************************************
************ present expert system reorg submenu.
choice = 0
do while choice # 3
    clear
    @ 2,1 say "RESEARCH MUSICOLOGICAL TOOL"
    @ 2,60 say dtoc(date()) + "    " + time()
    @ 3,0 to 3,79
    ?
    text
                EXPERT SYSTEM RULES REORGANIZATION

                Available reorganization features
include:

                1.  Set precedence of rule types
                2.  Set precedence of rule within rule
types
                3.  Exit to previous screen
     endtext
     @ 22,1 say "Enter choice (1 - 3) " get choice;
      picture "9" range 1,7
    read
*--------------------------------branch to appropriate
program.
    if choice < 1 .or. choice > 3
       @ 23,1 say "CHOICE MUST BE WITHIN RANGE OF 1-3"
       @ 24,1 say "PRESS ANY KEY TO CONTINUE"
       read
    else
       do case
           case choice = 1
               do prerule1
           case choice = 2
               do prerule2
       endcase
```

```
    endif (choice < 1 .or. choice > 3)

enddo (while choice # 3)
return


*********************************************************
***
*                          ESUTIL
*
*********************************************************
***

PROCEDURE ESUTIL
*****************************************************
********** expert system utilities subsystem******
*****************************************************
*----------------Create underline variable, uline.
************ present expert system maintenance submenu.
choice = 0
do while choice # 3
    clear
    @ 2,1 say "RESEARCH MUSICOLOGICAL TOOL"
    @ 2,60 say dtoc(date()) + "    " + time()
    @ 3,0 to 3,79
    ?
    text
                EXPERT SYSTEM UTILITIES

                Available expert system utilities
include:

                1.  Match an incipit
                2.  Query all similar incipits
                3.  Exit to previous screen
    endtext
    @ 20,1 say "Enter choice (1-3) " get choice;
      picture "9" range 1,3
    read
*--------------------------------branch to appropriate
program.
    if choice < 1 .or. choice > 3
       @ 23,1 say "CHOICE MUST BE WITHIN RANGE OF 1-3"
       @ 24,1 say "PRESS ANY KEY TO CONTINUE"
       read
    else
       do case
           case choice = 1
```

```
            do esmatch
        case choice = 2
            do esquery
    endcase
endif (choice < 1 .or. choice > 3)

enddo (while choice # 3)
close databases
return


*********************************************************
***
*                          INCCADD
*
*********************************************************
***

PROCEDURE INCCADD
*-------------------------------------------------------
------
parameters old_call, m_compl, m_compf, m_compi, c_link
************************************
********** incipit/composer add****
************************************
clear
*----------------Init memory variables
************ present composer add subsystem.
@ 2,1 say "RESEARCH MUSICOLOGICAL TOOL"
@ 2,60 say dtoc(date()) + "    " + time()
@ 3,0 to 3,79
@ 5,10 say "    DATABASE SYSTEM:  ADD COMPOSER
INFORMATION"
m_cloc = space(40)
m_bdate = space(4)
m_ddate = space(4)
@ 08,1 say "COMPOSER INFORMATION"
@ 10,01 say "Last name:" get m_compl picture "@!"
@ 11,01 say "First name:" get m_compf picture "@!"
@ 12,01 say "Initial:" get m_compi picture "@!"
read
error = .f.
if m_compl = space(40)
  error = .t.
endif (m_compl = space(40)
if .not. error
****before you do this must have opened the database,
****look for entered composer
```

```
select 3
use composer index c_lname
seek m_compl
if  found()
  if comp_init = m_compi .and. comp_fname = m_compf
    @ 20,1 clear
    @ 21,1 clear
    @ 20,1 say "This composer already exists on database"
    store .t. to m_ans
    @ 21,1 say "Do you wish to associate him/her ;
with this composition?"    get m_ans picture "Y"
    read
    if m_ans
      select 4
      use written index callw, cnamew
      append blank
      replace call_num with old_call
      replace comp_lname with m_compl
      replace comp_fname with m_compf
      replace comp_init with m_compi
      c_link = .t.
      select 3
    endif (m_ans)

  endif (comp_init = m_compi .and. comp_fname = m_compf)
endif (found())
select 3
if .not. found()
  @ 13,1 say "Birth year:"
  @ 13,15 get m_bdate
  @ 14,1 say "Death year:"
  @ 14,15 get m_ddate
  @ 15,1 say "Birthplace:"
  @ 15,15 get m_cloc picture "@!"
  error = .f.
  if val(m_bdate) > 1989
     error = .t.
  endif (bdate checks)
  if (val(m_ddate) > val(m_bdate) .or. val(m_ddate) >
1989) ;
     .and. m_ddate <> space(4)
     error = .t.
  endif (ddate checks)
  if .not. error
  read
  append blank
  replace comp_fname with m_compf
  replace comp_init with m_compi
```

```
  replace comp_lname with m_compl
  replace comp_bdate with m_bdate
  replace comp_ddate with m_ddate
  replace comp_loc with m_cloc
  select 4
  use written index callw, cnamew
  append blank
  replace call_num with old_call
  replace comp_lname with m_compl
  replace comp_fname with m_compf
  replace comp_init with m_compi
  c_link = .t.
  select 3
  else
    @ 22,1 say "Error in date entry."
    m_ans = .t.
    @ 23,1 say "Do you want to try again(Y/N)?" get m_ans
;
      picture "Y"
    read
    if .not. m_ans
      error = .t.
    endif(.not. m_ans)
  endif (.not. error)
 else
  if comp_init <> m_compi .or. comp_fname <> m_compf
    @ 13,1 say "Birth year:"
    @ 13,15 get m_bdate
    @ 14,1 say "Death year:"
    @ 14,15 get m_ddate
    @ 15,1 say "Birthplace:"
    @ 15,15 get m_cloc picture "@!"
    read
     error = .f.
     if val(m_bdate) > 1989
        error = .t.
     endif (bdate checks)
     if (val(m_ddate) < val(m_bdate) .or. val(m_ddate) >
1989) ;
        .and. m_ddate <> spaces
        error = .t.
     endif (ddate checks)
     if .not. error
    append blank
    replace comp_fname with m_compf
    replace comp_init with m_compi
    replace comp_lname with m_compl
    replace comp_bdate with m_bdate
```

```
    replace comp_ddate with m_ddate
    replace comp_loc with m_cloc
    select 4
    use written index callw, cnamew
    replace call_num with old_call
    replace comp_lname with m_compl
    replace comp_fname with m_compf
    replace comp_init with m_compi
    c_link = .t.
    select 3
     else
      @ 22,1 say "Error in date entry."
      m_ans = .t.
      @ 23,1 say "Do you want to try again(Y/N)?" get
m_ans ;
        picture "Y"
      read
      if .not. m_ans
         error = .t.
      endif (.not. m_ans)
     endif (.not. error)
  endif (comp_init = m_compi .and. comp_fname = m_compf)
endif (.not. found())
endif (.not. error)
return
*-----------------------------------------------------------
```

```
*************************************************************
***
*                              INCSADD
*
*************************************************************
***

PROCEDURE INCSADD
*-------------------------------------------------------------
parameters old_call, m_stitle, s_link

*************************************
********** incipit/source add****
*************************************
clear
*-----------------Init memory variables
************ present rule type add subsystem.
@ 2,1 say "RESEARCH MUSICOLOGICAL TOOL"
@ 2,60 say dtoc(date()) + "    " + time()
@ 3,0 to 3,79
@ 5,10 say "    DATABASE SYSTEM:  ADD MUSIC SOURCE"
select 5
use source index sources
m_stype = space(40)
m_sdate = space(4)
@ 08,1 say "SOURCE INFORMATION"
@ 10,1 say "Enter source:" get m_stitle picture "@!"
read
error = .f.
if m_stitle = space(40)
   error = .t.
endif (m_stitle = space(40)
if .not. error
****before you do this must have opened the database,
****look for entered source
seek m_stitle
if  found()
    store src_title to m_stitle
    @ 20,1 clear
    @ 21,1 clear
    @ 20,1 say "This source already exists"
    store .t. to m_ans
    @ 21,1 say "Do you wish to link it to this
composition?" ;
    get m_ans picture "Y"
    read
    if m_ans
```

```
      select 6
      use refer_in index callr, srcr
      append blank
      replace call_num with old_call
      replace src_title with m_stitle
      s_link = .t.
      select 5
    endif (m_ans)
endif (found())
select 5
if .not. found()
  @ 11,1 say "Source type:"
  @ 11,15 get m_stype  picture "@!"
  @ 12,1 say "Source date:"
  @ 12,15 get m_sdate
  read
  error = .f.
  if .not. (val(m_sdate) <= 1989)
    error = .t.
  endif (m_sdate check)
  if .not. error
  append blank
  replace src_type with m_stype
  replace src_title with m_stitle
  replace src_date with m_sdate
  select 6
  use refer_in index callr, srcr
  append blank
  replace call_num with old_call
  replace src_title with m_stitle
  s_link = .t.
  select 5
  else
   @ 22,1 say "Error in date entry."
   m_ans = .t.
   @ 23,1 say "Do you wish to try again(Y/N)?" get m_ans
;
   picture "Y"
   read
   if .not. m_ans
      error = .f.
   endif (.not. m_ans)
  endif (.not. error)
endif (.not. found())
endif (.not. error)
return
****************
```

```
*********************************************************
***
*                          INFER
*
*********************************************************
***

PROCEDURE INFER
parameters m_d1, m_p1, m_a1, ;
m_d2, m_p2, m_a2, ;
m_d3, m_p3, m_a3, ;
m_d4, m_p4, m_a4, ;
m_d5, m_p5, m_a5, ;
m_d6, m_p6, m_a6, ;
m_d7, m_p7, m_a7, ;
m_d8, m_p8, m_a8, ;
m_d9, m_p9, m_a9, ;
m_d10, m_p10, m_a10, ;
okflag, m_call, difcter, m_time1, m_time2
newdif = 0
************************************************
********** checks for inference engine  *********
************************************************
select 1
use music index call_num
reindex
if .not. BOF()
   go top
   @ 18,1 say "Please wait while the expert checks the
following:"
   do while .not. EOF()
*---init all variables---
    newdif = 0
*** how many differences in notes?
    notedif = 0
*** for check to see if time sig is multiple of itself
*** possibilities: "S" = same, "M" = multiple, "D" =
different
    toptime = "S"
    lowtime = "S"
    timedif = 0
*** how many differences in rhythm
    rhydif = 0
*** check for transposition??
    store .f. to trans
    store .f. to sameflag
    c_p1 = m_p1
```

```
    c_p2 = m_p2
    c_p3 = m_p3
    c_p4 = m_p4
    c_p5 = m_p5
    c_p6 = m_p6
    c_p7 = m_p7
    c_p8 = m_p8
    c_p9 = m_p9
    c_p10 = m_p10
    do chgnotes with m_p1, m_a1, m_p2, m_a2, m_p3, m_a3,
;
m_p4, m_a4, m_p5, m_a5, m_p6, m_a6, m_p7, m_a7, m_p8,
m_a8, m_p9, ;
m_a9, m_p10, m_a10, c_p1, c_p2, c_p3, ;
c_p4, c_p5, c_p6, c_p7, c_p8, c_p9, c_p10

************* checks ************

     do notes with  c_p1, c_p2, c_p3, c_p4, c_p5, ;
c_p6, c_p7, c_p8, c_p9, c_p10, notedif
     do rhythm with  m_d1, m_d2, m_d3, m_d4, m_d5, ;
m_d6, m_d7, m_d8, m_d9, m_d10, rhydif
     do time with m_time1, m_time2, toptime, lowtime,
timedif
*    do transp with m_d1, c_p1, m_d2, c_p2, m_d3, c_p3, ;
*m_d4, c_p4, m_d5, c_p5, m_d6, c_p6, m_d7, c_p7, m_d8, ;
*c_p8, m_d9, c_p9, m_d10, c_p10, trans
*     if notedif = 0 .and. rhydif = 0
*       if timedif = 0
*          sameflag = .t.
*       else
*          if trans
*            if timedif = 0
*               sameflag = .t.
*            endif (timedif = 0)
*          endif (trans)
*        endif (timedif = 0)
*       endif (notedif = 0 .and. rhydif = 0)
*       if sameflag
*        okflag = .f.
*       else
          do rcheck with notedif, rhydif, lowtime,
toptime, newdif
          select 1
          if newdif > difcter
           store newdif to difcter
          endif (newdif > difcter)
*       endif (sameflag )
```

```
        if .not. EOF()
        skip
        endif (.not. EOF())
     enddo (.not. EOF())
endif (.not. BOF())
m_ans = .t.
if difcter > 5
   okflag = .f.
endif (difcter > 5)
if .not. okflag
   clrcter = 18
   do while clrcter < 24
      @ clrcter,0 clear
      clrcter = clrcter + 1
   enddo (clrcter < 24)
   @ 21,1 say "Based on an evaluation of the rules, the
expert "
   @ 22,1 say "recommends that you NOT add this incipit."
   @ 23,1 say "Do you want to override this
recommendation(Y/N)?" ;
     get m_ans picture "Y"
     read
   if m_ans
      okflag = .t.
   endif (m_ans)
endif (.not. okflag)
clrcter = 18
do while clrcter < 24
   @ clrcter,0 clear
   clrcter = clrcter + 1
enddo (clrcter < 24)
return
```

```
*************************************************************
***
*                              MENU
*
*************************************************************
***

PROCEDURE MENU
**************************************
********** user frontend menu*********
**************************************
set help off
set talk off
set status off
set score off
set safety off
set bell off
set exact on
*-----------------Create underline variable, uline.
************ present user frontend menu and get user's
choice.
choice = 0
do while choice # 3
    clear
    @ 2,1 say "RESEARCH MUSICOLOGICAL TOOL"
    @ 2,60 say dtoc(date()) + "    " + time()
    @ 3,0 to 3,79
    ?
    ?
    text
                PRIMARY MENU

                Available selections include:


                1.  Database Facility
                2.  Expert System Maintenance
                3.  Exit Research Musicological Tool
    endtext
    @ 20,1 say "Enter choice " get choice;
      picture "9" range 1,3
    read
*---------------------------------branch to appropriate
program.
    if choice < 1 .or. choice > 3
       @ 23,1 say "CHOICE MUST BE WITHIN RANGE OF 1-3 "
       @ 24,1 say "PRESS ANY KEY TO CONTINUE"
```

```
      read
   else
      do case
          case choice = 1
              do dbfac
          case choice = 2
              do esmain
      endcase
   endif (choice < 1 .or. choice > 3)
enddo (while choice # 3)
close databases
clear
*quit


*************************************************************
***
*                          NOTES
*
*************************************************************
***

PROCEDURE NOTES
parameters c_p1, c_p2, c_p3, c_p4, ;
c_p5, c_p6, c_p7, c_p8, c_p9, ;
c_p10, notedif
x_p1 = inc_p1
x_p2 = inc_p2
x_p3 = inc_p3
x_p4 = inc_p4
x_p5 = inc_p5
x_p6 = inc_p6
x_p7 = inc_p7
x_p8 = inc_p8
x_p9 = inc_p9
x_p10 = inc_p10
do chgnote2 with inc_p1, inc_a1, inc_p2, inc_a2, inc_p3,
;
inc_a3, inc_p4, inc_a4, inc_p5, inc_a5, inc_p6, inc_a6, ;
inc_p7, inc_a7, inc_p8, inc_a8, inc_p9, inc_a9, inc_p10,
;
inc_a10, x_p1, x_p2, x_p3, x_p4, x_p5, x_p6, x_p7, x_p8,
;
x_p9, x_p10
if x_p1 <> c_p1
    notedif = notedif + 1
endif (x_p1 <> c_p1)
if x_p2 <> c_p2
```

```
   notedif = notedif + 1
endif (x_p2 <> c_p2)
if x_p3 <> c_p3
   notedif = notedif + 1
endif (x_p3 <> c_p3)
if x_p4 <> c_p4
   notedif = notedif + 1
endif (x_p4 <> c_p4)
if x_p5 <> c_p5
   notedif = notedif + 1
endif (x_p5 <> c_p5)
if x_p6 <> c_p6
   notedif = notedif + 1
endif (x_p6 <> c_p6)
if x_p7 <> c_p7
   notedif = notedif + 1
endif (x_p7 <> c_p7)
if x_p8 <> c_p8
   notedif = notedif + 1
endif (x_p8 <> c_p8)
if x_p9 <> c_p9
   notedif = notedif + 1
endif (x_p9 <> c_p9)
if x_p10 <> c_p10
   notedif = notedif + 1
endif (x_p10 <> c_p10)
@ 20,1 clear
@ 21,1 clear
@ 20,1 say "NOTES"
? "Difference in notes is ", notedif
return
```

```
*************************************************************
***
*                          PRERULE1
*
*************************************************************
***

PROCEDURE PRERULE1
*****************************************
********** rule type reorder subsystem***
*****************************************
more = .t.
do while more
clear
*----------------Init memory variables
************ present rule type reorder subsystem***
@ 2,1 say "RESEARCH MUSICOLOGICAL TOOL"
@ 2,60 say dtoc(date()) + "    " + time()
@ 3,0  to 3,79
@ 5,10 say "    EXPERT SYSTEM UTILITY:  REORDER RULE
TYPE"
m_type = space(20)
m_seq_num1 = 000
@ 08,1 say "Enter rule type you wish to reorder:";
  get m_type picture "@!"
read
****before you do this must open the database,
****look for entered rule
select 7
use rule index types, num1, num2
reindex
seek m_type
if found()
  @ 09,1 say "This rule type is currently ordered as
number "
  @ 09,48 say seq_num1
  store seq_num1 to o_seq_num
  @ 10,1 say "Change this order to:" get m_seq_num1
  read
  error = .f.
***need to know last seq_num1***
  do while .not. EOF()
    skip
  enddo(.not. EOF())
  if BOF()
    store 0 to last_one
  else
```

```
    skip -1
    store seq_num1 to last_one
  endif (BOF())
  if m_seq_num1 > last_one
     error = .t.
  endif (m_seq_num1 > last_one)
 if .not. error
  store .f. to m_ans
  @ 19,1 say "Make this change?" get m_ans picture "Y"
  read
  if m_ans
    if m_seq_num1 < o_seq_num
      replace seq_num1 with m_seq_num1 for type = m_type
      replace seq_num1 with seq_num1 + 1 for ;
     .not. (type = m_type .or. seq_num1 < m_seq_num1 ;
     .or. seq_num1 > o_seq_num)
    else
      if m_seq_num1 > o_seq_num
        replace seq_num1 with m_seq_num1 for type =
m_type
        replace seq_num1 with seq_num1 - 1 for ;
       .not. (type = m_type .or. seq_num1 > m_seq_num1
;
        .or. seq_num1 < o_seq_num)
      endif (m_seq_num1 > o_seq_num)
    endif (m_seq_num1 < o_seq_num)
  endif (m_ans)
  answer = .t.
  @ 22,1 say "Do you want to reorder another rule type?"
;
    get answer picture "Y"
  read
  if .not. answer
   more = .f.
  endif (answer)
else
  @ 21,1 clear
  ? "New sequence number cannot exceed " ,last_one
  m_ans = .t.
  @ 23,1 say "Do you want to try again(Y/N)?" get m_ans ;
  picture "Y"
  read
  if .not. m_ans
     more = .f.
  endif (.not. m_ans)
 endif (.not. error)
else
  m_ans = .t.
```

```
  @ 22,1 say "This rule does not exist."
  @ 23,1 say "Do you want to try another (Y/N)?:" get
m_ans ;
    picture "Y"
  read
  if .not. m_ans
     more = .f.
  endif (.not. m_ans)
endif (found())
enddo (do while more)
close databases
return


*********************************************************
***
*                          PRERULE2
*
*********************************************************
***

PROCEDURE PRERULE2
*****************************************
********** rule reorder subsystem***
*****************************************
more = .t.
do while more
clear
*-----------------Init memory variables
************* present rule within type reorder
subsystem***
@ 2,1 say "RESEARCH MUSICOLOGICAL TOOL"
@ 2,60 say dtoc(date()) + "    " + time()
@ 3,0 to 3,79
@ 5,10 say  " EXPERT SYSTEM UTILITY:  REORDER RULES
WITHIN TYPES"
m_type = space(20)
m_seq_num2 = 000
m_field1 = space(39)
m_oper = space(2)
m_field2 = space(39)
@ 08,1 say "Enter rule type you wish to reorder:";
  get m_type picture "@!"
read
****before you do this must open the database,
****look for entered rule
select 7
use rule index types, num1, num2
```

```
reindex
seek m_type
if found()
   @ 09,1 say "Change the order of what rule?"
   @ 10,1 say "Enter Rule Field1  :" get m_field1 picture
"@!"
   @ 11,1 say "      Rule Operator:" get m_oper picture
"@!"
   @ 12,1 say "      Rule Field2  :" get m_field2 picture
"@!"
   read
   ds_rule = 0

   do while type = m_type .and. .not. EOF()
     if field1 = m_field1 .and. oper = m_oper .and.
field2 ;
        = m_field2
       store seq_num2 to ds_rule
       store seq_num2 to o_seq_num2
     endif (matches the rule parameters)
     skip
   enddo
   skip -1
   store seq_num2 to last_one
   rule_fd = .t.
   if ds_rule = 0
     ****must act like not found***
     rule_fd = .f.
   endif (ds_rule = 0)
   if rule_fd
   @ 14,1 clear
   ? " This is currently rule " , ds_rule
   ? " Rules of this type total", last_one
   m_ans = .t.
   @ 17,1 say "Do you wish to change the order of this
rule?" ;
     get m_ans picture "Y"
    read
   if m_ans
   @ 18,1 say "New order of this rule is: " get
m_seq_num2
   read
   error = .f.
   if m_seq_num2 > last_one .or. m_seq_num2 < 1
      error = .t.
   endif (m_seq_num2 > last_one .or. m_seq_num2 < 1)
   if .not. error
     if m_seq_num2 < o_seq_num2
```

```
       use rule index num2, num1, types
       seek o_seq_num2
       replace all seq_num2 with m_seq_num2 for type =
m_type ;
.and. seq_num2 = o_seq_num2
       use rule index types, num1, num2
       replace all seq_num2 with seq_num2 + 1 for type =
m_type ;
.and. (seq_num2 > m_seq_num2 .or. (seq_num2 = m_seq_num2
;
.and. (field1 <> m_field1  ;
.or. oper <> m_oper .or. field2 <> m_field2))) .and. ;
(seq_num2 = o_seq_num2 .or. seq_num2 < o_seq_num2)
    else
      if m_seq_num2 > o_seq_num2
        use rule index num2, num1, types
         seek o_seq_num2
         replace all seq_num2 with m_seq_num2 for type =
m_type ;
.and. seq_num2 = o_seq_num2
         use rule index types, num1, num2
         replace all seq_num2 with seq_num2 - 1 for type
= m_type ;
.and. (seq_num2 < m_seq_num2 .or. (seq_num2 = m_seq_num2
;
.and. (field1 <> m_field1 ;
.or. oper <> m_oper .or. field2 <> m_field2))) .and. ;
(seq_num2 >= o_seq_num2) .and. ;
seq_num2 <> 0
      endif (m_seq_num2 < o_seq_num2)
    endif (m_seq_num2 > o_seq_num2)

answer = .t.
@ 22,1 say "Do you want to reorder another rule type?" ;
     get answer picture "Y"
read
if .not.answer
   more = .f.
endif (.not. answer)
else
  @ 21,1 clear
  ? " New sequence number must be > 1 and < " , last_one
  m_ans = .t.
  @ 23,1 say "Do you want to try again(Y/N)?" get m_ans ;
  picture "Y"
  read
  if .not. m_ans
     more = .f.
```

```
  endif (.not. m_ans)
endif (.not. error)
endif (m_ans)
else
  @ 22,1 say "This rule does not exist."
  m_ans = .t.
  @ 23,1 say "Do you want to try again(Y/N)?" get m_ans ;
    picture "Y"
  read
  if .not. m_ans
     more = .f.
  endif (.not. m_ans)

endif (rule_fd)
else
   @ 22,1 say "This rule type does not exist."
   m_ans = .t.
   @ 23,1 say "Do you want to try again (Y/N)?" get m_ans
;
     picture "Y"
   read
  if .not. m_ans
     more = .f.
  endif (m_ans)
endif (.not. found)
enddo (do while more)
close databases
return


*************************************************************
***
*                            RCHECK
*
*************************************************************
***

PROCEDURE RCHECK
parameters notedif, rhydif, toptime, lowtime, newdif
select 7
use rule index num1, num2, types
reindex
if .not. BOF()
   @ 20,0 clear
   @ 20,1 say "RULE CHECKS"
   go top
   do while .not. EOF()
    if seq_num2 > 0 .and. act_flag1 .and. act_flag2
```

```
          macfld1 = field1
          macfld2 = oper
          macfld3 = field2
          @ 22,0 clear
          @ 22,1 say macfld1
          @ 22,15 say macfld2
          @ 22, 20 say macfld3
          if .not. (&macfld1 &macfld2 &macfld3)
            if seq_num2 = 1
              newdif = newdif + 5
            else
              newdif = newdif + 1
            endif (seq_num2 = 1)
            @ 22,40 say "STATUS = FAILED"
          else
             @ 22,40 say "STATUS = PASSED"
          endif (&macfld1 &macfld2 &macfld3)
      endif (seq_num2 > 0  .and. act_flag1 .and.
act_flag2)
      if .not. EOF()
      skip
      endif (.not. EOF())
    enddo (.not. EOF())
endif (.not. BOF())
return


*************************************************************
***
*                            RHYTHM
*
*************************************************************
***

PROCEDURE RHYTHM
parameters m_d1, m_d2, m_d3, m_d4, ;
m_d5, m_d6, m_d7, m_d8, m_d9, ;
m_d10, rhydif
if inc_d1 <> m_d1
    rhydif = rhydif + 1
endif (inc_d1 <> m_d1)
if inc_d2 <> m_d2
   rhydif = rhydif + 1
endif (inc_d2 <> m_d2)
if inc_d3 <> m_d3
   rhydif = rhydif + 1
endif (inc_d3 <> m_d3)
if inc_d4 <> m_d4
```

```
   rhydif = rhydif + 1
endif (inc_d4 <> m_d4)
if inc_d5 <> m_d5
   rhydif = rhydif + 1
endif (inc_d5 <> m_d5)
if inc_d6 <> m_d6
   rhydif = rhydif + 1
endif (inc_d6 <> m_d6)
if inc_d7 <> m_d7
   rhydif = rhydif + 1
endif (inc_d7 <> m_d7)
if inc_d8 <> m_d8
   rhydif = rhydif + 1
endif (inc_d8 <> m_d8)
if inc_d9 <> m_d9
   rhydif = rhydif + 1
endif (inc_d9 <> m_d9)
if inc_d10 <> m_d10
   rhydif = rhydif + 1
endif (inc_d10 <> m_d10)
@ 20,1 clear
@ 21,1 clear
@ 20,1 say "RHYTHM"
? "Difference in rhythm is ", rhydif
return


************************************************************
***
*                              RPNOTES
*
************************************************************
***

PROCEDURE RPNOTES
*------------------------------------------------------------
-------
parameters m_d1, m_p1, m_a1, ;
m_d2, m_p2, m_a2, ;
m_d3, m_p3, m_a3, ;
m_d4, m_p4, m_a4, ;
m_d5, m_p5, m_a5, ;
m_d6, m_p6, m_a6, ;
m_d7, m_p7, m_a7, ;
m_d8, m_p8, m_a8, ;
m_d9, m_p9, m_a9, ;
m_d10, m_p10, m_a10
```

```
*-------this replaces fields in the music database with
those --
*-------in the temporary graphf database----------------
--------
*-----------------------------------------------------
--------
*--------used by addinc program-------------------------
--------
*-----------------------------------------------------
--------
replace inc_p1 with m_p1
replace inc_d1 with m_d1
replace inc_a1 with m_a1
replace inc_d2 with m_d2
replace inc_p2 with m_p2
replace inc_a2 with m_a2
replace inc_d3 with m_d3
replace inc_p3 with m_p3
replace inc_a3 with m_a3
replace inc_d4 with m_d4
replace inc_p4 with m_p4
replace inc_a4 with m_a4
replace inc_d5 with m_d5
replace inc_p5 with m_p5
replace inc_a5 with m_a5
replace inc_d6 with m_d6
replace inc_p6 with m_p6
replace inc_a6 with m_a6
replace inc_d7 with m_d7
replace inc_p7 with m_p7
replace inc_a7 with m_a7
replace inc_d8 with m_d8
replace inc_p8 with m_p8
replace inc_a8 with m_a8
replace inc_d9 with m_d9
replace inc_p9 with m_p9
replace inc_a9 with m_a9
replace inc_d10 with m_d10
replace inc_p10 with m_p10
replace inc_a10 with m_a10
return
*----------end of storage routine------------------
```

```
*************************************************************
***
*                              STNOTES
*
*************************************************************
***

PROCEDURE STNOTES
*-----------------------------------------
parameters m_d1, m_p1, m_a1, ;
m_d2, m_p2, m_a2, ;
m_d3, m_p3, m_a3, ;
m_d4, m_p4, m_a4, ;
m_d5, m_p5, m_a5, ;
m_d6, m_p6, m_a6, ;
m_d7, m_p7, m_a7, ;
m_d8, m_p8, m_a8, ;
m_d9, m_p9, m_a9, ;
m_d10, m_p10, m_a10
*-------this reads from the graphf database and store to
memory--
*-------------------------------------------------------
--------
*--------used by addinc program-------------------------
-------
*-------------------------------------------------------
-------
store inc_d1 to m_d1
store inc_p1 to m_p1
store inc_a1 to m_a1
store inc_d2 to m_d2
store inc_p2 to m_p2
store inc_a2 to m_a2
store inc_d3 to m_d3
store inc_p3 to m_p3
store inc_a3 to m_a3
store inc_d4 to m_d4
store inc_p4 to m_p4
store inc_a4 to m_a4
store inc_d5 to m_d5
store inc_p5 to m_p5
store inc_a5 to m_a5
store inc_d6 to m_d6
store inc_p6 to m_p6
store inc_a6 to m_a6
store inc_d7 to m_d7
store inc_p7 to m_p7
```

```
store inc_a7 to m_a7
store inc_d8 to m_d8
store inc_p8 to m_p8
store inc_a8 to m_a8
store inc_d9 to m_d9
store inc_p9 to m_p9
store inc_a9 to m_a9
store inc_d10 to m_d10
store inc_p10 to m_p10
store inc_a10 to m_a10
return
*----------end of storage routine------------------


*********************************************************
***
*                            TIME
*
*********************************************************
***

PROCEDURE TIME
parameters m_time1, m_time2, toptime, lowtime, timedif
if time1 = m_time1
   toptime = "S"
else
   if mod(time1,m_time1) = 0
      toptime = "M"
   else
      toptime = "D"
   endif (mod(time1,mtime1) = 0)
endif (time1 = m_time1)
if time2 = m_time2
   lowtime = "S"
else
   if mod(time2,m_time2) = 0
      lowtime = "M"
   else
      lowtime = "D"
   endif (mod(time2,m_time2) = 0)
endif (time2 = m_time2)
if toptime = "S" .and. lowtime = "S"
   timedif = 0
else
   if toptime = "M" .and. lowtime = "M"
      timedif = 0
   else
      timedif = 1
```

```
   endif (toptime = "M" .and. lowtime = "M")
endif (toptime = "S" .and. lowtime = "S")

@ 20,1 clear
@ 21,1 clear
@ 20,1 say "TIME"
? "Difference in time is ", timedif
return


*********************************************************
***
*                          TITLES
*
*********************************************************
***

PROCEDURE TITLES
parameters title
@ 2,1 say "RESEARCH MUSICOLOGICAL TOOL"
@ 2,60 say dtoc(date()) + "    " + time()
@ 3,0 to 3,79
@5,10 say title
return


*********************************************************
***
*                          TRANSP
*
*********************************************************
***

PROCEDURE TRANSP
parameters m_d1, c_p1, m_d2, c_p2, m_d3, c_p3, m_d4,
c_p4, ;
m_d5, c_p5, m_d6, c_p6, m_d7, c_p7, m_d8, c_p8, m_d9,
c_p9, ;
m_d10, c_p10, trans
@ 20,1 clear
@ 20,1 say "TRANSPOSITION"
@ 21,0 clear
@ 22,0 clear
return
```

```
*********************************************************
***
*                          UPDCOMP
*
*********************************************************
***

PROCEDURE UPDCOMP
**************************************
*********** composer update subsystem**
**************************************
more = .t.
do while more
clear
*----------------Init memory variables
************ present composer update subsystem.
@ 2,1 say "RESEARCH MUSICOLOGICAL TOOL"
@ 2,60 say dtoc(date()) + "    " + time()
@ 3,0 to 3,79
@ 5,10 say "    DATABASE SYSTEM:  UPDATE COMPOSER
INFORMATION"
m_compf = space(10)
m_compi = space(1)
m_compl = space(20)
@ 08,1 say "COMPOSER INFORMATION"
@ 10,1 say "Last name:" get m_compl picture "@!"
@ 11,1 say "First name:" get m_compf picture "@!"
@ 12,1 say "Initial:" get m_compi picture "@!"
read
use composer index c_lname
seek m_compl
if .not. found()
    @ 22,1 say "This composer does not exist on database"
    m_ans = .t.
    @ 23,1 say "Do you want to try again(Y/N)?" get m_ans
;
    picture "Y"
    read
    if .not. m_ans
       more = .f.
    endif (.not. m_ans)
endif (.not. found())
if found()
   if comp_init = m_compi .and. comp_fname = m_compf
     store comp_bdate to m_bdate
     store comp_ddate to m_ddate
```

```
     store comp_loc to m_cloc
     @ 13,1 say "Birth date:"
     @ 13,15 get m_bdate
     @ 14,1 say "Death date:"
     @ 14,15 get m_ddate
     @ 15,1 say "Birthplace:"
     @ 15,15 get m_cloc picture "@!"
     read
     error = .f.
     if val(m_bdate) > 1989
        error = .t.
     endif (bdate checks)
     if (val(m_ddate) < val(m_bdate) .or. val(m_ddate) >
1989) ;
.and. m_ddate <> space(4)
       error = .t.
     endif (ddate checks)
     if .not. error
     store .f. to m_ans
     @ 19,1 say "Make these changes?" get m_ans picture
"Y"
     read
     if m_ans
       replace comp_bdate with m_bdate
       replace comp_ddate with m_ddate
       replace comp_loc with m_cloc
     endif (m_ans)
     answer = .t.
     @ 23,1 say "Do you want to update another composer?"
;
         get answer picture "Y"
     read
     if .not. answer
       more = .f.
     endif (.not. answer)
    else
      @ 22,1 clear
      @ 22,1 say "Invalid date entry"
      m_ans = .t.
      @ 23,1 clear
      @ 23,1 say "Do you want to try again(Y/N)?" get
m_ans ;
        picture "Y"
        read
      if .not. m_ans
         more = .f.
      endif (.not. m_ans)
    endif (.not. error)
```

```
   else
      @ 22,1 say "This composer does not exist on
database."
      m_ans = .t.
      @ 23,1 say "Do you want to try again(Y/N)?" get
m_ans ;
      picture "Y"
      read
      if .not. m_ans
         more = .f.
      endif (.not. m_ans)
   endif (comp_init = m_compi .and. comp_fname = m_compf)
endif ( found())
enddo (do while more)
return
```

```
*********************************************************
***
*                          UPDINC
*
*********************************************************
***

PROCEDURE UPDINC
********* database incipit update****
more = .t.
do while more
clear
*-----------------Init memory variables
*---in order to delete a music record---
*------1  schedule graphics interface with  call #
*------    and activity indicator  - "U" for update
*------2  graphics interface update incipit
*------3  double check that they want this update
*------4  if update to incipit is desired and ok return
*--------    from graphics interface
*-------5   schedule inference engine
*---------else
*------------try another call number???
*-----------------------------------------------------
--------
m_call = 00000
new_num = 00000
************* present incipit update subsystem.
@ 2,1 say "RESEARCH MUSICOLOGICAL TOOL"
@ 2,60 say dtoc(date()) + "    " + time()
@ 3,0 to 3,79
@ 5,10 say "    DATABASE SYSTEM:  UPDATE MUSIC RECORD"
m_d1 = 0
m_p1 = 0
m_a1 = " "
m_d2 = 0
m_p2 = 0
m_a2 = " "
m_d3 = 0
m_p3 = 0
m_a3 = " "
m_d4 = 0
m_p4 = 0
m_a4 = " "
m_d5 = 0
m_p5 = 0
m_a5 = " "
```

```
m_d6 = 0
m_p6 = 0
m_a6 = " "
m_d7 = 0
m_p7 = 0
m_a7 = " "
m_d8 = 0
m_p8 = 0
m_a8 = " "
m_d9 = 0
m_p9 = 0
m_a9 = " "
m_d10 = 0
m_p10 = 0
m_a10 = " "
select 1
use music index call_num
@ 10,1 say "Enter the call number you wish to update:" ;
  get m_call picture '99999'
read
seek m_call
if .not. found()
   @ 22,1 say "This call number does not exist"
   m_ans = .t.
   @ 23,1 say "Do you want to try again(Y/N)?" get m_ans
;
   picture "Y"
   read
   if .not. m_ans
     more = .f.
   endif (.not. m_ans)
endif (.not. found())
if found()
* erase \project\gfile1.doc
  erase \tp\gfile1.doc
  store title to m_title
  store time1 to m_time1
  store time2 to m_time2
  store key to m_key
  store mm to m_mm
  do stnotes with m_d1, m_p1, m_a1, ;
m_d2, m_p2, m_a2, ;
m_d3, m_p3, m_a3, ;
m_d4, m_p4, m_a4, ;
m_d5, m_p5, m_a5, ;
m_d6, m_p6, m_a6, ;
m_d7, m_p7, m_a7, ;
m_d8, m_p8, m_a8, ;
```

```
m_d9, m_p9, m_a9, ;
m_d10, m_p10, m_a10
   @ 22,1 say "Please wait to update incipit"
   select 2
   use graphf index callg
   append blank
   replace act with "U"
   new_num = m_call + 10000
   replace call with new_num
   replace key with m_key
   replace mm with m_mm
   replace time1 with m_time1
   replace time2 with m_time2
   do rpnotes with m_d1, m_p1, m_a1, ;
m_d2, m_p2, m_a2, ;
m_d3, m_p3, m_a3, ;
m_d4, m_p4, m_a4, ;
m_d5, m_p5, m_a5, ;
m_d6, m_p6, m_a6, ;
m_d7, m_p7, m_a7, ;
m_d8, m_p8, m_a8, ;
m_d9, m_p9, m_a9, ;
m_d10, m_p10, m_a10
*----send file to graphic interface-----
* copy to \project\gfile1.doc sdf
  copy to \tp\gfile1.doc sdf
zap
run cd \tp
run grapintf
run cd \demo
*-----SIMULATE GRAPHICS INTERFACE HERE
title = "    DATABASE SYSTEM:  UPDATE MUSIC RECORD"
do titles with title
@ 22,1 clear
store .t. to m_ans
@ 22,1 say "Make these updates?" ;
get m_ans picture "Y"
read
if m_ans
  select 2
  use graphf index callg
* append from gfile2.doc sdf
  append from \tp\gifile.dat sdf
  new_num = 10000 - call
  store new_num to m_call
  store mm to m_mm
  store m_time1 to m_time1
  store m_time2 to m_time2
```

```
  store key to m_key
  do stnotes with m_d1, m_p1, m_a1, ;
m_d2, m_p2, m_a2, ;
m_d3, m_p3, m_a3, ;
m_d4, m_p4, m_a4, ;
m_d5, m_p5, m_a5, ;
m_d6, m_p6, m_a6, ;
m_d7, m_p7, m_a7, ;
m_d8, m_p8, m_a8, ;
m_d9, m_p9, m_a9, ;
m_d10, m_p10, m_a10
  if act = "U"
    okupd = .t.
    store 0 to difcter
    do infer with m_d1, m_p1, m_a1, ;
m_d2, m_p2, m_a2, ;
m_d3, m_p3, m_a3, ;
m_d4, m_p4, m_a4, ;
m_d5, m_p5, m_a5, ;
m_d6, m_p6, m_a6, ;
m_d7, m_p7, m_a7, ;
m_d8, m_p8, m_a8, ;
m_d9, m_p9, m_a9, ;
m_d10, m_p10, m_a10, ;
okupd, m_call, difcter, m_time1, m_time2
    if okupd
      select 1
      use music index call_num
      seek m_call
      if found()
         replace key with m_key
         replace title with m_title
         replace time1 with m_time1
         replace time2 with m_time2
         do rpnotes with m_d1, m_p1, m_a1, ;
m_d2, m_p2, m_a2, ;
m_d3, m_p3, m_a3, ;
m_d4, m_p4, m_a4, ;
m_d5, m_p5, m_a5, ;
m_d6, m_p6, m_a6, ;
m_d7, m_p7, m_a7, ;
m_d8, m_p8, m_a8, ;
m_d9, m_p9, m_a9, ;
m_d10, m_p10, m_a10
     endif (found())
  else
   @ 22,1 say "Update is not indicated by graphics
interface"
```

```
   endif (okupd)
  endif (act = "U")
 endif (m_ans)
answer = .t.
 @ 22,1 say "Do you want to update another incipit?" ;
 get answer picture "Y"
 read
 if .not. answer
   more = .f.
 endif (.not. answer)
endif (found())
select 2
use graphf index callg
zap
enddo (more)
close databases
return


************************************************************
***
*                              UPDSRC
*
************************************************************
***

PROCEDURE UPDSRC
**************************************
*********** source update subsystem***
**************************************
more = .t.
do while more
clear
*-----------------Init memory variables
************ present source update subsystem.
@ 2,1 say "RESEARCH MUSICOLOGICAL TOOL"
@ 2,60 say dtoc(date()) + "    " + time()
@ 3,0  to 3,79
@ 5,10 say "    DATABASE SYSTEM:  UPDATE MUSIC SOURCE"
use source index sources
m_stitle = space(40)
@ 08,1 say "SOURCE INFORMATION"
@ 10,1 say "Enter source:" get m_stitle picture "@!"
read
****before you do this must have opened the database,
****look for entered source
seek m_stitle
if .not. found()
```

```
    @ 22,1 say "This source does not exist"
    m_ans = .t.
    @ 23,1 say "Do you want to try again(Y/N)?" get m_ans
;
    picture "Y"
    read
    if .not. m_ans
      more = .f.
    endif (.not. m_ans)
endif (.not. found())
if found()
    store src_type to m_stype
    store src_date to m_sdate
    store src_title to m_stitle
  @ 11,1 say "Source type:"
  @ 11,15 get m_stype picture "@!"
  @ 12,1 say "Source date:"
  @ 12,15 get m_sdate
  read
  error = .f.
  if val(m_sdate) > 1989
     error = .t.
  endif (date check)
  if .not. error
  store .f. to m_ans
  @ 19,1 say "Make these changes?" get m_ans picture "Y"
  read
  if m_ans
    replace src_type with m_stype
    replace src_date with m_sdate
  endif (m_ans )
  answer = .t.
  @ 23,1 say "Do you want to change another source?" ;
     get answer picture "Y"
  read
  if .not. answer
    more = .f.
  endif (.not. answer)
  else
    m_ans = .t.
    @ 22,1 say "Invalid date entry."
    @ 23,1 say "Do you want to try again(Y/N)?" get m_ans
;
      picture "Y"
    read
    if .not. m_ans
       more = .f.
    endif (.not. m_ans)
```

```
  endif (.not. error)
endif (.not. found())
enddo (more)
return
```

A MUSICOLOGICAL RESEARCH TOOL: AN EXPERT SYSTEM SOLUTION FOR SMALL PROJECTS

by

JEANNINE STAFFORD INGRAM

B.S., University of North Carolina-Greensboro, 1982

AN ABSTRACT OF A REPORT

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Computing and Information Sciences

KANSAS STATE UNIVERSITY
Manhattan, Kansas

1989

Abstract

Musicology, the scholarly study of music, was first introduced as a discipline by Friedrich Chrysander in 1863. In the past 125 years, this discipline has become one of sound historical research techniques. It requires that the professional musicologist have a broad knowledge base and be able to organize and assimilate vast quantities of data, all of which may play some part in the final research product. Among the many areas within the humanities, musicology perhaps would most benefit from a single, integrated system capable of assisting in the completion of a multi-faceted project.

The primary motivation of this project was to provide a PC-based automated research tool for musicologists and small music archives and to explore the usefulness of expert systems as a potential research tool for music historians.

The project will establish the need for a research musicological tool, develop an expert system capable of incipit matching and decision processing within the environment of standard musicological research techniques, and develop a database management system suitable for the special needs of small projects in musicology.